wpa_supplicant/hostapd

0.7.x

Generated by Doxygen 1.6.1

Sat Nov 28 23:07:45 2009

Contents

Chapter 1

Developers' documentation for wpa_supplicant and hostapd

The goal of this documentation and comments in the source code is to give enough information for other developers to understand how wpa_supplicant and hostapd have been implemented, how they can be modified, how new drivers can be supported, and how the source code can be ported to other operating systems. If any information is missing, feel free to contact Jouni Malinen <j@w1.fi> for more information. Contributions as patch files are also very welcome at the same address. Please note that this software is licensed under dual license, GPLv2 or BSD at user's choice. All contributions to wpa_supplicant and hostapd are expected to use compatible licensing terms.

The source code and read-only access to the combined wpa_supplicant and hostapd Git repository is available from the project home page at http://w1.fi/wpa_supplicant/. This developers' documentation is also available as a PDF file from http://w1.fi/wpa_supplicant/wpa_supplicant-devel.pdf .

1.1 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 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++.

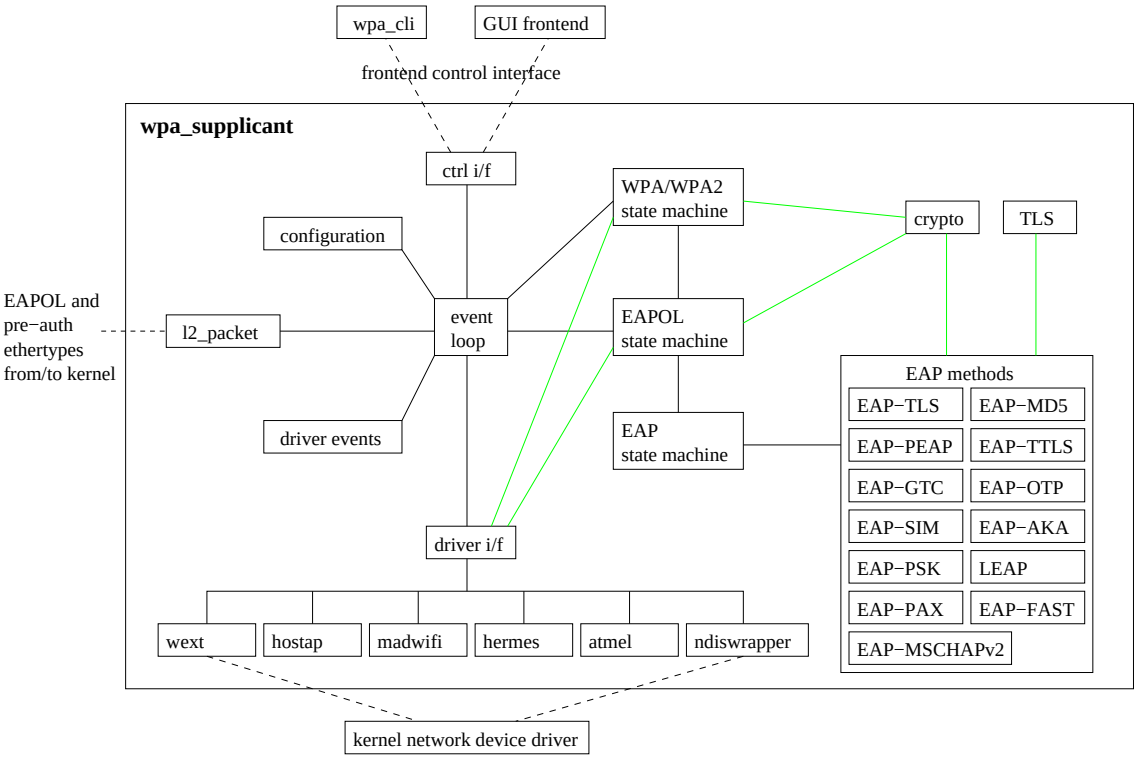


Figure 1.1: wpa_supplicant modules

1.2 hostapd

hostapd includes IEEE 802.11 access point management (authentication / association), IEEE 802.1X/WPA/WPA2 Authenticator, EAP server, and RADIUS authentication server functionality. It can be build with various configuration option, e.g., a standalone AP management solution or a RADIUS authentication server with support for number of EAP methods.

The design goal for hostapd 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 server implementation. Similarly, RADIUS authentication server is in its own separate module. Both IEEE 802.1X and RADIUS authentication server can use EAP server functionality.

hostapd implements a control interface that can be used by external programs to control the operations of the hostapdt 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++.

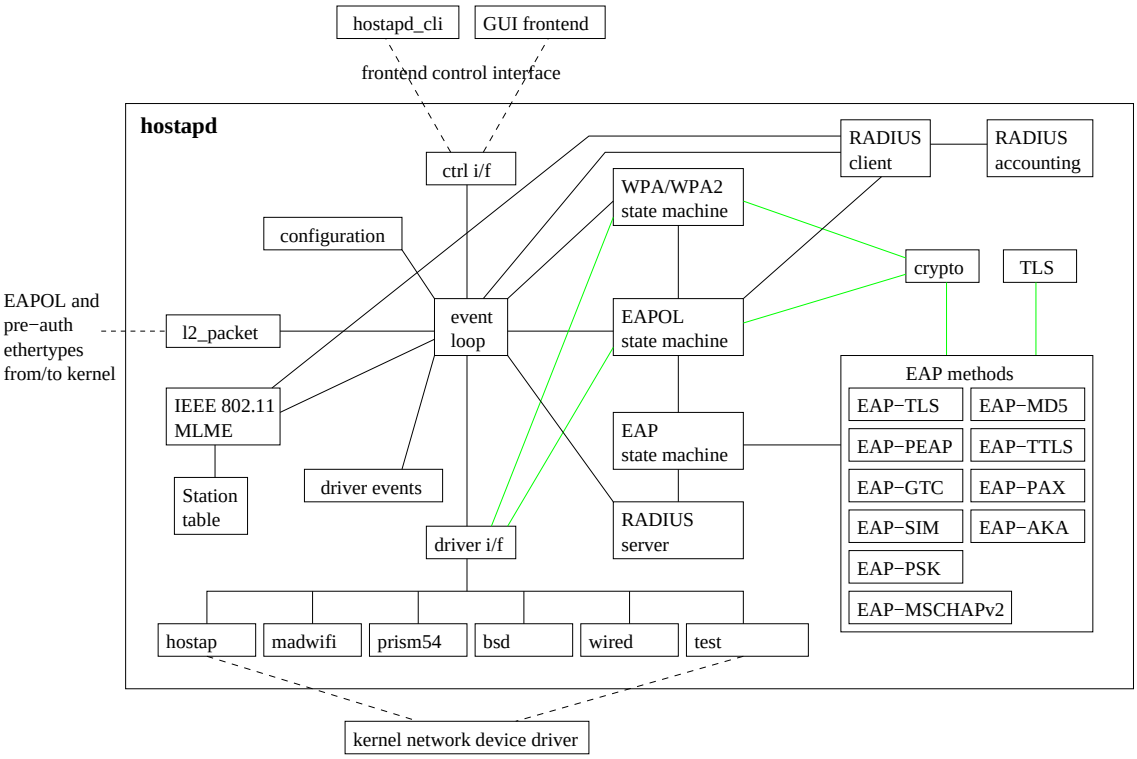


Figure 1.2: hostapd modules

Chapter 2

Structure of the source code

[wpa_supplicant core functionality | Generic helper functions | Cryptographic functions | TLS library | 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.

2.1 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

2.2 Generic helper functions

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

eloop.c and eloop.h Event loop (select() loop with registerable timeouts, socket read callbacks, and signal callbacks)

common.c and common.h Common helper functions

defs.h Definitions shared by multiple files

l2_packet.h, l2_packet_linux.c, and l2_packet_pcap.c Layer 2 (link) access wrapper (includes native Linux implementation and wrappers for libdnet/libpcap). A new l2_packet implementation may need to be added when porting to new operating systems that are not supported by libdnet/libpcap. Makefile can be used to select which l2_packet implementation is included. l2_packet_linux.c uses Linux packet sockets and l2_packet_pcap.c has a more portable version using libpcap and libdnet.

pcsc_funcs.c and pcsc_funcs.h Wrapper for PC/SC lite SIM and smart card readers

priv_netlink.h Private version of netlink definitions from Linux kernel header files; this could be replaced with C library header file once suitable version becomes commonly available

version.h Version number definitions

wireless_copy.h Private version of Linux wireless extensions definitions from kernel header files; this could be replaced with C library header file once suitable version becomes commonly available

2.3 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)

sha256.c and sha256.h SHA-256 (replaced with a crypto library if TLS support is included)

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_openssl.c Wrapper functions for libcrypto (OpenSSL)

crypto_internal.c Wrapper functions for internal crypto implementation

crypto_gnutls.c Wrapper functions for libgcrypt (used by GnuTLS)

ms_funcs.c and ms_funcs.h Helper functions for MSCHAPV2 and LEAP

tls.h Definition of TLS library wrapper

tls_none.c Dummy implementation of TLS library wrapper for cases where TLS functionality is not included.

tls_openssl.c TLS library wrapper for openssl

tls_internal.c TLS library for internal TLS implementation

tls_gnutls.c TLS library wrapper for GnuTLS

2.4 TLS library

asn1.c and asn1.h ASN.1 DER parsing

bignum.c and bignum.h Big number math

rsa.c and rsa.h RSA

x509v3.c and x509v3.h X.509v3 certificate parsing and processing

tlsv1_client.c, tlsv1_client.h TLSv1 client (RFC 2246)

tlsv1_client_i.h Internal structures for TLSv1 client

tlsv1_client_read.c TLSv1 client: read handshake messages

tlsv1_client_write.c TLSv1 client: write handshake messages

tlsv1_common.c and tlsv1_common.h Common TLSv1 routines and definitions

tlsv1_cred.c and tlsv1_cred.h TLSv1 credentials

tlsv1_record.c and tlsv1_record.h TLSv1 record protocol

2.5 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)

config_winreg.c Configuration backend for Windows registry

2.6 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

ctrl_iface_unix.c UNIX domain sockets -based control interface backend

ctrl_iface_udp.c UDP sockets -based control interface backend

ctrl_iface_named_pipe.c Windows named pipes -based control interface backend

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

2.7 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.

2.8 EAP peer

EAP peer implementation is a separate module that can be used by other programs than just wpa_supplicant.

eap.c and eap.h EAP state machine and method interface

eap_defs.h Common EAP definitions

eap_i.h Internal definitions for EAP state machine and EAP methods; not to be included in other modules

eap_sim_common.c and eap_sim_common.h Common code for EAP-SIM and EAP-AKA

eap_tls_common.c and eap_tls_common.h Common code for EAP-PEAP, EAP-TTLS, and EAP-FAST

eap_tlv.c and eap_tlv.h EAP-TLV code for EAP-PEAP and EAP-FAST

eap_ttls.c and eap_ttls.h EAP-TTLS

eap_pax.c, eap_pax_common.h, eap_pax_common.c EAP-PAX

eap_psk.c, eap_psk_common.h, eap_psk_common.c EAP-PSK (note: this is not needed for WPA-PSK)

eap_sake.c, eap_sake_common.h, eap_sake_common.c EAP-SAKE

eap_gpsk.c, eap_gpsk_common.h, eap_gpsk_common.c EAP-GPSK

eap_aka.c, eap_fast.c, eap_gtc.c, eap_leap.c, eap_md5.c, eap_mschapv2.c, eap_otp.c, eap_peap.c, eap_-sim.c, eap_tls.c Other EAP method implementations

2.9 EAPOL supplicant

eapol_supp_sm.c and eapol_supp_sm.h EAPOL supplicant state machine and IEEE 802.1X processing

2.10 Windows port

ndis_events.c Code for receiving NdisMIndicateStatus() events and delivering them to wpa_supplicant driver_ndis.c in more easier to use form

win_if_list.c External program for listing current network interface

2.11 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

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

Chapter 3

wpa_supplicant 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.

3.1 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().

3.2 Control interface commands

Following commands can be used with wpa_ctrl_request():

3.2.1 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.

3.2.2 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
```

3.2.3 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
```

3.2.4 STATUS-VERBOSE

Same as STATUS, but with more verbosity (i.e., more `variable=value` pairs).

```
bssid=02:00:01:02:03:04
ssid=test network
id=0
```

```
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA-PSK
wpa_state=COMPLETED
ip_address=192.168.1.21
Supplicant PAE state=AUTHENTICATED
suppPortStatus=Authorized
heldPeriod=60
authPeriod=30
startPeriod=30
maxStart=3
portControl=Auto
Supplicant Backend state=IDLE
EAP state=SUCCESS
reqMethod=0
methodState=NONE
decision=COND_SUCC
ClientTimeout=60
```

3.2.5 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
```

3.2.6 SET <variable> <value>

Set variables:

- EAPOL::heldPeriod
- EAPOL::authPeriod
- EAPOL::startPeriod
- EAPOL::maxStart
- dot11RSNAConfigPMKLifetime
- dot11RSNAConfigPMKReauthThreshold
- dot11RSNAConfigSATimeout

Example command:

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

3.2.7 LOGON

IEEE 802.1X EAPOL state machine logon.

3.2.8 LOGOFF

IEEE 802.1X EAPOL state machine logoff.

3.2.9 REASSOCIATE

Force reassociation.

3.2.10 RECONNECT

Connect if disconnected (i.e., like `REASSOCIATE`, but only connect if in disconnected state).

3.2.11 PREAUTH <BSSID>

Start pre-authentication with the given BSSID.

3.2.12 ATTACH

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

3.2.13 DETACH

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

3.2.14 LEVEL <debug level>

Change debug level.

3.2.15 RECONFIGURE

Force wpa_supplicant to re-read its configuration data.

3.2.16 TERMINATE

Terminate wpa_supplicant process.

3.2.17 BSSID <network id> <BSSID>

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

3.2.18 LIST_NETWORKS

List configured networks.

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

(note: fields are separated with tabs)

3.2.19 DISCONNECT

Disconnect and wait for REASSOCIATE or RECONNECT command before connecting.

3.2.20 SCAN

Request a new BSS scan.

3.2.21 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)

3.2.22 BSS

Get detailed per-BSS scan results. BSS command can be used to iterate through scan results one BSS at a time and to fetch all information from the found BSSes. This provides access to the same data that is available through SCAN_RESULTS but in a way that avoids problems with large number of scan results not fitting in the ctrl_iface messages.

There are two options for selecting the BSS with the BSS command: "BSS <idx>" requests information for the BSS identified by the index (0 .. size-1) in the scan results table and "BSS <BSSID>" requests information for the given BSS (based on BSSID in 00:01:02:03:04:05 format).

BSS information is presented in following format. Please note that new fields may be added to this field=value data, so the ctrl_iface user should be prepared to ignore values it does not understand.

```
bssid=00:09:5b:95:e0:4e
freq=2412
beacon_int=0
capabilities=0x0011
qual=51
noise=161
level=212
tsf=0000000000000000
ie=000b6a6b6d207072697661746501018dd180050f20101000050f20401000050f20401000050f2020000
ssid=jkm private
```

3.2.23 SELECT_NETWORK <network id>

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

3.2.24 ENABLE_NETWORK <network id>

Enable a network. Network id can be received from the LIST_NETWORKS command output. Special network id all can be used to enable all network.

3.2.25 DISABLE_NETWORK <network id>

Disable a network. Network id can be received from the LIST_NETWORKS command output. Special network id all can be used to disable all network.

3.2.26 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.

3.2.27 REMOVE_NETWORK <network id>

Remove a network. Network id can be received from the LIST_NETWORKS command output. Special network id all can be used to remove all network.

3.2.28 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)
- ...

3.2.29 GET_NETWORK <network id> <variable>

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

3.2.30 SAVE_CONFIG

Save the current configuration.

3.3 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
```

3.3.1 GET_CAPABILITY <option> [strict]

Get list of supported functionality (eap, pairwise, group, proto). Supported functionality is shown as space separate lists of values used in the same format as in wpa_supplicant configuration. If optional argument, 'strict', is added, only the values that the driver claims to explicitly support are included. Without this, all available capabilities are included if the driver does not provide a mechanism for querying capabilities.

Example request/reply pairs:

```
GET_CAPABILITY eap
AKA FAST GTC LEAP MD5 MSCHAPV2 OTP PAX PEAP PSK SIM TLS TTLS

GET_CAPABILITY pairwise
CCMP TKIP NONE

GET_CAPABILITY pairwise strict

GET_CAPABILITY group
CCMP TKIP WEP104 WEP40

GET_CAPABILITY key_mgmt
WPA-PSK WPA-EAP IEEE8021X NONE

GET_CAPABILITY proto
RSN WPA

GET_CAPABILITY auth_alg
OPEN SHARED LEAP
```

3.3.2 AP_SCAN <ap_scan value>

Change ap_scan value: 0 = no scanning, 1 = wpa_supplicant requests scans and uses scan results to select the AP, 2 = wpa_supplicant does not use scanning and just requests driver to associate and take care of AP selection

3.3.3 INTERFACES

List configured interfaces.

```
wlan0
eth0
```

Chapter 4

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 driver.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.

4.1 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.

4.1.1 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 functional, 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 implementa-

tions 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.

4.1.2 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)

4.1.3 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 driver.h). wpa_-supplicant is normally configured to use ap_scan=2 mode with drivers that control WPA IE generation and roaming.

4.1.4 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 driver.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.

4.1.5 Summary of Linux Wireless Extensions use

AP selection depends on ap_scan configuration:

ap_scan=1:

- wpa_supplicant requests scan with SIOCSIWSCAN
- driver reports scan complete with wireless event SIOCGIWSCAN
- wpa_supplicant reads scan results with SIOCGIWSCAN (multiple call if a larget buffer is needed)
- wpa_supplicant decides which AP to use based on scan results
- wpa_supplicant configures driver to associate with the selected BSS (SIOCSIWMODE, SIOCSIWGENIE, SIOCSIWAUTH, SIOCSIWFREQ, SIOCSIWESSID, SIOCSIWAP)

ap_scan=2:

- wpa_supplicant configures driver to associate with an SSID (SIOCSIWMODE, SIOCSIWGENIE, SIOCSIWAUTH, SIOCSIWESSID)

After this, both modes use similar steps:

- optionally (or required for drivers that generate WPA/RSN IE for (Re)AssocReq), driver reports association parameters (AssocReq IEs) with wireless event IWEVASSOCREQIE (and optionally IWEVASSOCRESPIE)
- driver reports association with wireless event SIOCGIWAP
- wpa_supplicant takes care of EAPOL frame handling (validating information from associnfo and if needed, from scan results if WPA/RSN IE from the Beacon frame is not reported through associnfo)

Chapter 5

EAP peer implementation

Extensible Authentication Protocol (EAP) is an authentication framework defined in RFC 3748. wpa_supplicant uses a separate code module for EAP peer implementation. This module was designed to use only a minimal set of direct function calls (mainly, to debug/event functions) in order for it to be usable in other programs. The design of the EAP implementation is based loosely on RFC 4137. The state machine is defined in this RFC and so is the interface between the peer state machine and methods. As such, this RFC provides useful information for understanding the EAP peer implementation in wpa_supplicant.

Some of the terminology used in EAP state machine is referring to EAPOL (IEEE 802.1X), but there is no strict requirement on the lower layer being IEEE 802.1X if EAP module is built for other programs than wpa_supplicant. These terms should be understood to refer to the lower layer as defined in RFC 4137.

5.1 Adding EAP methods

Each EAP method is implemented as a separate module, usually as one C file named eap_<name of the method>.c, e.g., eap_md5.c. All EAP methods use the same interface between the peer state machine and method specific functions. This allows new EAP methods to be added without modifying the core EAP state machine implementation.

New EAP methods need to be registered by adding them into the build (Makefile) and the EAP method registration list in the eap_peer_register_methods() function of eap_methods.c. Each EAP method should use a build-time configuration option, e.g., EAP_TLS, in order to make it possible to select which of the methods are included in the build.

EAP methods must implement the interface defined in eap_i.h. struct eap_method defines the needed function pointers that each EAP method must provide. In addition, the EAP type and name are registered using this structure. This interface is based on section 4.4 of RFC 4137.

It is recommended that the EAP methods would use generic helper functions, eap_msg_alloc() and eap_hdr_validate() when processing messages. This allows code sharing and can avoid missing some of the needed validation steps for received packets. In addition, these functions make it easier to change between expanded and legacy EAP header, if needed.

When adding an EAP method that uses a vendor specific EAP type (Expanded Type as defined in RFC 3748, Chapter 5.7), the new method must be registered by passing vendor id instead of EAP_VENDOR_IETF to eap_peer_method_alloc(). These methods must not try to emulate expanded types by registering a legacy EAP method for type 254. See eap_vendor_test.c for an example of an EAP method implementation that is implemented as an expanded type.

5.2 Using EAP implementation as a library

The Git repository has an eap_example directory that contains an example showing how EAP peer and server code from wpa_supplicant and hostapd can be used as a library. The example program initializes both an EAP server and an EAP peer entities and then runs through an EAP-PEAP/MSCHAPv2 authentication.

eap_example_peer.c shows the initialization and glue code needed to control the EAP peer implementation. eap_example_server.c does the same for EAP server. eap_example.c is an example that ties in both the EAP server and client parts to allow an EAP authentication to be shown.

In this example, the EAP messages are passed between the server and the peer are passed by direct function calls within the same process. In practice, server and peer functionalities would likely reside in separate devices and the EAP messages would be transmitted between the devices based on an external protocol. For example, in IEEE 802.11 uses IEEE 802.1X EAPOL state machines to control the transmission of EAP messages and WiMax supports optional PMK EAP authentication mechanism that transmits EAP messages

as defined in IEEE 802.16e.

The EAP library links in number of helper functions from src/utils and src/crypto directories. Most of these are suitable as-is, but it may be desirable to replace the debug output code in src/utils/wpa_debug.c by dropping this file from the library and re-implementing the functions there in a way that better fits in with the main application.

Chapter 6

EAP server implementation

Extensible Authentication Protocol (EAP) is an authentication framework defined in RFC 3748. hostapd uses a separate code module for EAP server 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 server state machine and methods. As such, this RFC provides useful information for understanding the EAP server implementation in hostapd.

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.

6.1 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 server state machine and method specific functions. This allows new EAP methods to be added without modifying the core EAP state machine implementation.

New EAP methods need to be registered by adding them into the build (Makefile) and the EAP method registration list in the eap_server_register_methods() function of eap_methods.c. Each EAP method should use a build-time configuration option, e.g., EAP_TLS, in order to make it possible to select which of the methods are included in the build.

EAP methods must implement the interface defined in eap_i.h. struct eap_method defines the needed function pointers that each EAP method must provide. In addition, the EAP type and name are registered using this structure. This interface is based on section 4.4 of RFC 4137.

It is recommended that the EAP methods would use generic helper functions, eap_msg_alloc() and eap_hdr_validate() when processing messages. This allows code sharing and can avoid missing some of the needed validation steps for received packets. In addition, these functions make it easier to change between expanded and legacy EAP header, if needed.

When adding an EAP method that uses a vendor specific EAP type (Expanded Type as defined in RFC 3748, Chapter 5.7), the new method must be registered by passing vendor id instead of EAP_VENDOR_IETF to eap_server_method_alloc(). These methods must not try to emulate expanded types by registering a legacy EAP method for type 254. See eap_vendor_test.c for an example of an EAP method implementation that is implemented as an expanded type.

Chapter 7

hostapd control interface

hostapd implements a control interface that can be used by external programs to control the operations of the hostapd 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++. hostapd_cli.c is an example program using this library.

There are multiple mechanisms for inter-process communication. For example, Linux version of hostapd is using UNIX domain sockets for the control interface. The use of the functions defined in wpa_ctrl.h can be used to hide the details of the used IPC from external programs.

7.1 Using the control interface

External programs, e.g., a GUI or a configuration utility, that need to communicate with hostapd 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().

hostapd 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 hostapd. These can be executed using wpa_ctrl_request(). Unsolicited event messages are sent by hostapd 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 Control interface commands

Following commands can be used with wpa_ctrl_request():

7.2.1 PING

This command can be used to test whether hostapd is replying to the control interface commands. The expected reply is `PONG` if the connection is open and hostapd is processing commands.

Chapter 8

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, the code has been ported to number of other operating systems like VxWorks, PalmOS, Windows CE, and Windows Mobile. On the hardware side, wpa_supplicant is used on various systems: desktops, laptops, PDAs, and embedded devices with CPUs including x86, PowerPC, arm/xscale, and MIPS. Both big and little endian configurations are supported.

8.1 Extra functions on top of ANSI C

wpa_supplicant is mostly using ANSI C functions that are available on most targets. However, couple of additional functions that are common on modern UNIX systems are used. Number of these are listed with prototypes in common.h (the

```
#ifdef CONFIG_ANSI_C_EXTRA
```

block). These functions may need to be implemented or at least defined as macros to native functions in the target OS or C library.

Many of the common ANSI C functions are used through a wrapper definitions in os.h to allow these to be replaced easily with a platform specific version in case standard C libraries are not available. In addition, os.h defines couple of common platform specific functions that are implemented in os_unix.c for UNIX like targets and in os_win32.c for Win32 API. If the target platform does not support either of these examples, a new os_*.c file may need to be added.

Unless OS_NO_C_LIB_DEFINES is defined, the standard ANSI C and POSIX functions are used by defining the os_*() wrappers to use them directly in order to avoid extra cost in size and speed. If the target platform needs different versions of the functions, os.h can be modified to define the suitable macros or alternatively, OS_NO_C_LIB_DEFINES may be defined for the build and the wrapper functions can then be implemented in a new os_*.c wrapper file.

common.h defines number of helper macros for handling integers of different size and byte order. Suitable version of these definitions may need to be added for the target platform.

8.2 Configuration backend

wpa_supplicant implements a configuration interface that allows the backend to be easily replaced in order to read configuration data from a suitable source depending on the target platform. config.c implements the generic code that can be shared with all configuration backends. Each backend is implemented in its own config_*.c file.

The included config_file.c backend uses a text file for configuration and config_winreg.c uses Windows registry. These files can be used as an example for a new configuration backend if the target platform uses different mechanism for configuration parameters. In addition, config_none.c can be used as an empty starting point for building a new configuration backend.

8.3 Driver interface

Unless the target OS and driver is already supported, most porting projects have to implement a driver wrapper. This may be done by adding a new driver interface module or modifying an existing module (driver_*.c) if the new target is similar to one of them. Driver wrapper implementation describes the details of the driver interface and discusses the tasks involved in porting this part of wpa_supplicant.

8.4 l2_packet (link layer access)

wpa_supplicant needs to have access to sending and receiving layer 2 (link layer) packets with two Ethertypes: EAP-over-LAN (EAPOL) 0x888e and RSN pre-authentication 0x88c7. l2_packet.h defines the interfaces used for this in the core wpa_supplicant implementation.

If the target operating system supports a generic mechanism for link layer access, that is likely the best mechanism for providing the needed functionality for wpa_supplicant. Linux packet socket is an example of such a generic mechanism. If this is not available, a separate interface may need to be implemented to the network stack or driver. This is usually an intermediate or protocol driver that is operating between the device driver and the OS network stack. If such a mechanism is not feasible, the interface can also be implemented directly in the device driver.

The main wpa_supplicant repository includes l2_packet implementations for Linux using packet sockets (l2_packet_linux.c), more portable version using libpcap/libdnet libraries (l2_packet_pcap.c; this supports WinPcap, too), and FreeBSD specific version of libpcap interface (l2_packet_freebsd.c).

If the target operating system is supported by libpcap (receiving) and libdnet (sending), l2_packet_pcap.c can likely be used with minimal or no changes. If this is not a case or a proprietary interface for link layer is required, a new l2_packet module may need to be added. Alternatively, struct wpa_driver_ops::send_eapol() handler can be used to override the l2_packet library if the link layer access is integrated with the driver interface implementation.

8.5 Event loop

wpa_supplicant uses a single process/thread model and an event loop to provide callbacks on events (registered timeout, received packet, signal). eloop.h defines the event loop interface. eloop.c is an implementation of such an event loop using select() and sockets. This is suitable for most UNIX/POSIX systems. When porting to other operating systems, it may be necessary to replace that implementation with OS specific mechanisms that provide similar functionality.

8.6 Control interface

wpa_supplicant uses a control interface to allow external processed to get status information and to control the operations. Currently, this is implemented with socket based communication; both UNIX domain sockets and UDP sockets are supported. If the target OS does not support sockets, this interface will likely need to be modified to use another mechanism like message queues. The control interface is optional component, so it is also possible to run wpa_supplicant without porting this part.

The wpa_supplicant side of the control interface is implemented in ctrl_iface.c. Matching client side is implemented as a control interface library in wpa_ctrl.c.

8.7 Program entry point

wpa_supplicant defines a set of functions that can be used to initialize main supplicant processing. Each operating system has a mechanism for starting new processing or threads. This is usually a function with a specific set of arguments and calling convention. This function is responsible on initializing wpa_supplicant.

main.c includes an entry point for UNIX-like operating system, i.e., main() function that uses command line arguments for setting parameters for wpa_supplicant. When porting to other operating systems, similar

OS-specific entry point implementation is needed. It can be implemented in a new file that is then linked with wpa_supplicant instead of main.o. main.c is also a good example on how the initialization process should be done.

The supplicant initialization functions are defined in wpa_supplicant_i.h. In most cases, the entry point function should start by fetching configuration parameters. After this, a global wpa_supplicant context is initialized with a call to wpa_supplicant_init(). After this, existing network interfaces can be added with wpa_supplicant_add_iface(). wpa_supplicant_run() is then used to start the main event loop. Once this returns at program termination time, wpa_supplicant_deinit() is used to release global context data.

wpa_supplicant_add_iface() and wpa_supplicant_remove_iface() can be used dynamically to add and remove interfaces based on when wpa_supplicant processing is needed for them. This can be done, e.g., when hotplug network adapters are being inserted and ejected. It is also possible to do this when a network interface is being enabled/disabled if it is desirable that wpa_supplicant processing for the interface is fully enabled/disabled at the same time.

8.8 Simple build example

One way to start a porting project is to begin with a very simple build of wpa_supplicant with WPA-PSK support and once that is building correctly, start adding features.

Following command can be used to build very simple version of wpa_supplicant:

```
cc -o wpa_supplicant config.c eloop.c common.c md5.c rc4.c sha1.c \
config_none.c l2_packet_none.c tls_none.c wpa.c preauth.c \
aes_wrap.c wpa_supplicant.c events.c main_none.c drivers.c
```

The end result is not really very useful since it uses empty functions for configuration parsing and layer 2 packet access and does not include a driver interface. However, this is a good starting point since the build is complete in the sense that all functions are present and this is easy to configure to a build system by just including the listed C files.

Once this version can be build successfully, the end result can be made functional by adding a proper program entry point (main∗.c), driver interface (driver_∗.c and matching CONFIG_DRIVER_∗ define for registration in drivers.c), configuration parser/writer (config_∗.c), and layer 2 packet access implementation (l2_packet_∗.c). After these components have been added, the end result should be a working WPA/WPA2-PSK enabled supplicant.

After the basic functionality has been verified to work, more features can be added by linking in more files and defining C pre-processor defines. Currently, the best source of information for what options are available and which files needs to be included is in the Makefile used for building the supplicant with make. Similar configuration will be needed for build systems that either use different type of make tool or a GUI-based project configuration.

Chapter 9

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.

9.1 eapol_test - EAP peer and RADIUS client testing

eapol_test is a program that links together the same EAP peer implementation that wpa_supplicant is using and the RADIUS authentication client code from hostapd. In addition, it has minimal glue code to combine these two components in similar ways to IEEE 802.1X/EAPOL Authenticator state machines. In other words, it integrates IEEE 802.1X Authenticator (normally, an access point) and IEEE 802.1X Supplicant (normally, a wireless client) together to generate a single program that can be used to test EAP methods without having to setup an access point and a wireless client.

The main uses for eapol_test are in interoperability testing of EAP methods against RADIUS servers and in development testing for new EAP methods. It can be easily used to automate EAP testing for interoperability and regression since the program can be run from shell scripts without require additional test components apart from a RADIUS server. For example, the automated EAP tests described in eap_testing.txt are implemented with eapol_test. Similarly, eapol_test could be used to implement an automated regression test suite for a RADIUS authentication server.

eapol_test uses the same build time configuration file, .config, as wpa_supplicant. This file is used to select which EAP methods are included in eapol_test. This program is not built with the default Makefile target, so a separate make command needs to be used to compile the tool:

```
make eapol_test
```

The resulting eapol_test binary has following command like options:

```
usage:
eapol_test [-nWS] -c<conf> [-a<AS IP>] [-p<AS port>] [-s<AS secret>] \
           [-r<count>] [-t<timeout>] [-C<Connect-Info>] \
           [-M<client MAC address>]
eapol_test scard
eapol_test sim <PIN> <num triplets> [debug]

options:
  -c<conf> = configuration file
  -a<AS IP> = IP address of the authentication server, default 127.0.0.1
  -p<AS port> = UDP port of the authentication server, default 1812
  -s<AS secret> = shared secret with the authentication server, default 'radius'
  -r<count> = number of re-authentications
  -W = wait for a control interface monitor before starting
  -S = save configuration after authentiation
  -n = no MPPE keys expected
  -t<timeout> = sets timeout in seconds (default: 30 s)
  -C<Connect-Info> = RADIUS Connect-Info (default: CONNECT 11Mbps 802.11b)
  -M<client MAC address> = Set own MAC address (Calling-Station-Id,
                           default: 02:00:00:00:00:01)
```

As an example,

```
eapol_test -ctest.conf -a127.0.0.1 -p1812 -ssecret -r1
```

tries to complete EAP authentication based on the network configuration from test.conf against the RADIUS server running on the local host. A re-authentication is triggered to test fast re-authentication. The configuration file uses the same format for network blocks as wpa_supplicant.

9.2 preauth_test - WPA2 pre-authentication and EAP peer testing

preauth_test is similar to eapol_test in the sense that in combines EAP peer implementation with something else, in this case, with WPA2 pre-authentication. This tool can be used to test pre-authentication based on the code that wpa_supplicant is using. As such, it tests both the wpa_supplicant implementation and the functionality of an access point.

preauth_test is built with:

```
make preauth_test
```

and it uses following command line arguments:

```
usage: preauth_test <conf> <target MAC address> <ifname>
```

For example,

```
preauth_test test.conf 02:11:22:33:44:55 eth0
```

would use network configuration from test.conf to try to complete pre-authentication with AP using BSSID 02:11:22:33:44:55. The pre-authentication packets would be sent using the eth0 interface.

9.3 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
```

9.4 Unit tests

Number of the components (.c files) used in wpa_supplicant define their own unit tests for automated validation of the basic functionality. Most of the tests for cryptographic algorithms are using standard test vectors to validate functionality. These tests can be useful especially when verifying port to a new CPU target.

In most cases, these tests are implemented in the end of the same file with functions that are normally commented out, but ca be included by defining a pre-processor variable when building the file separately. The details of the needed build options are included in the Makefile (test-∗ targets). All automated unit tests can be run with

```
make tests
```

This make target builds and runs each test and terminates with zero exit code if all tests were completed successfully.

Chapter 10

Directory Hierarchy

10.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 11

Data Structure Index

11.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 12

File Index

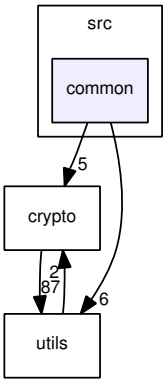
12.1 File List

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

Chapter 13

Directory Documentation

13.1 src/common/ Directory Reference



Files

- file defs.h

  WPA Supplicant - Common definitions.

- file eapol_common.h

  EAPOL definitions shared between hostapd and wpa_supplicant.

- file ieee802_11_common.c

  IEEE 802.11 Common routines.

- file ieee802_11_common.h

  IEEE 802.11 Common routines.

- file ieee802_11_defs.h

  IEEE 802.11 Frame type definitions.

- file **nl80211_copy.h**
- file privsep_commands.h

  WPA Supplicant - privilege separation commands.

- file **version.h**
- file **wireless_copy.h**
- file wpa_common.c

  WPA/RSN - Shared functions for supplicant and authenticator.

- file wpa_common.h

  WPA definitions shared between hostapd and wpa_supplicant.

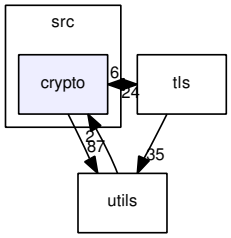
- file wpa_ctrl.c

  wpa_supplicant/hostapd control interface library

- file wpa_ctrl.h

  wpa_supplicant/hostapd control interface library

13.2 src/crypto/ Directory Reference



Files

- file aes-cbc.c
  AES-128 CBC.
- file aes-ctr.c
  AES-128 CTR.
- file aes-eax.c
  AES-128 EAX.
- file aes-encblock.c
  AES encrypt_block.
- file aes-internal-dec.c
  AES (Rijndael) cipher - decrypt.
- file aes-internal-enc.c
  AES (Rijndael) cipher - encrypt.
- file aes-internal.c
  AES (Rijndael) cipher.
- file aes-omac1.c
  One-key CBC MAC (OMAC1) hash with AES-128.
- file aes-unwrap.c
  AES key unwrap (128-bit KEK, RFC3394).
- file aes-wrap.c
  AES Key Wrap Algorithm (128-bit KEK) (RFC3394).
- file aes.h
  AES functions.
- file aes_i.h
  AES (Rijndael) cipher.

- file aes_wrap.h
  AES-based functions.
- file crypto.h
  WPA Supplicant / wrapper functions for crypto libraries.
- file crypto_cryptoapi.c
  WPA Supplicant / Crypto wrapper for Microsoft CryptoAPI.
- file crypto_gnutls.c
  WPA Supplicant / wrapper functions for libgcrypt.
- file crypto_internal.c
  WPA Supplicant / Crypto wrapper for internal crypto implementation.
- file crypto_libtomcrypt.c
  WPA Supplicant / Crypto wrapper for LibTomCrypt (for internal TLSv1).
- file crypto_none.c
  WPA Supplicant / Empty template functions for crypto wrapper.
- file crypto_nss.c
  Crypto wrapper functions for NSS.
- file crypto_openssl.c
  WPA Supplicant / wrapper functions for libcrypto.
- file des-internal.c
  DES and 3DES-EDE ciphers.
- file des_i.h
  DES and 3DES-EDE ciphers.
- file dh_group5.c
  Diffie-Hellman group 5 operations.
- file dh_group5.h
  Diffie-Hellman group 5 operations.
- file dh_groups.c
  Diffie-Hellman groups.
- file dh_groups.h
  Diffie-Hellman groups.
- file fips_prf_gnutls.c
  FIPS 186-2 PRF for libgcrypt.
- file fips_prf_internal.c

FIPS 186-2 PRF for internal crypto implementation.

- file fips_prf_nss.c

  FIPS 186-2 PRF for NSS.

- file fips_prf_openssl.c

  FIPS 186-2 PRF for libcrypto.

- file md4-internal.c

  MD4 hash implementation.

- file md5-internal.c

  MD5 hash implementation and interface functions.

- file md5-non-fips.c

  MD5 hash implementation and interface functions (non-FIPS allowed cases).

- file md5.c

  MD5 hash implementation and interface functions.

- file md5.h

  MD5 hash implementation and interface functions.

- file md5_i.h

  MD5 internal definitions.

- file ms_funcs.c

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

- file ms_funcs.h

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

- file rc4.c

  RC4 stream cipher.

- file sha1-internal.c

  SHA1 hash implementation and interface functions.

- file sha1-pbkdf2.c

  SHA1-based key derivation function (PBKDF2) for IEEE 802.11i.

- file sha1-tlsprf.c

  TLS PRF (SHA1 + MD5).

- file sha1-tprf.c

  SHA1 T-PRF for EAP-FAST.

- file sha1.c

  SHA1 hash implementation and interface functions.

- file sha1.h

  SHA1 hash implementation and interface functions.

- file sha1_i.h

  SHA1 internal definitions.

- file sha256-internal.c

  SHA-256 hash implementation and interface functions.

- file sha256.c

  SHA-256 hash implementation and interface functions.

- file sha256.h

  SHA256 hash implementation and interface functions.

- file tls.h

  WPA Supplicant / SSL/TLS interface definition.

- file tls_gnutls.c

  WPA Supplicant / SSL/TLS interface functions for openssl.

- file tls_internal.c

  WPA Supplicant / TLS interface functions and an internal TLS implementation.

- file tls_none.c

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

- file tls_nss.c

  SSL/TLS interface functions for NSS.

- file tls_openssl.c

  WPA Supplicant / SSL/TLS interface functions for openssl.

- file tls_schannel.c

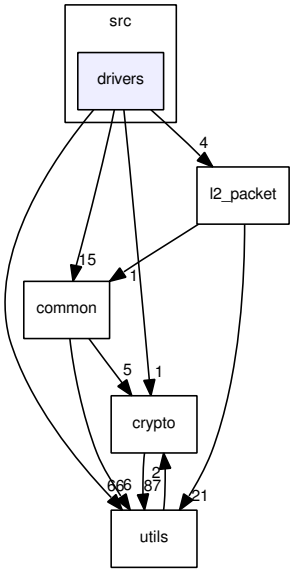
  WPA Supplicant / SSL/TLS interface functions for Microsoft Schannel.

13.3 doc/ Directory Reference

Files

- file **code_structure.doxygen**
- file **ctrl_iface.doxygen**
- file **driver_wrapper.doxygen**
- file **eap.doxygen**
- file **eap_server.doxygen**
- file **hostapd_ctrl_iface.doxygen**
- file **mainpage.doxygen**
- file **porting.doxygen**
- file **testing_tools.doxygen**

13.4 src/drivers/ Directory Reference



Files

- file **Apple80211.h**
- file driver.h

  WPA Supplicant - driver interface definition.

- file driver_atheros.c

  hostapd / Driver interaction with Atheros driver

- file driver_atmel.c

  WPA Supplicant - Driver interaction with Atmel Wireless LAN drivers.

- file driver_broadcom.c

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

- file driver_bsd.c

  WPA Supplicant - driver interaction with BSD net80211 layer.

- file driver_hostap.c

  Driver interaction with Linux Host AP driver.

- file driver_hostap.h

  Driver interaction with Linux Host AP driver.

- file driver_iphone.m

  WPA Supplicant - iPhone/iPod touch Apple80211 driver interface.

- file driver_ipw.c

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

- file driver_madwifi.c

  WPA Supplicant - driver interaction with MADWIFI 802.11 driver.

- file driver_ndis.c

  WPA Supplicant - Windows/NDIS driver interface.

- file driver_ndis.h

  WPA Supplicant - Windows/NDIS driver interface.

- file driver_ndis_.c

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

- file driver_ndiswrapper.c

  WPA Supplicant - driver interaction with Linux ndiswrapper.

- file driver_nl80211.c

  Driver interaction with Linux nl80211/cfg80211.

- file driver_none.c

  Driver interface for RADIUS server or WPS ER only (no driver).

- file driver_osx.m

  WPA Supplicant - Mac OS X Apple80211 driver interface.

- file driver_prism54.c

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

- file driver_privsep.c

  WPA Supplicant - privilege separated driver interface.

- file driver_ps3.c

  WPA Supplicant - PS3 Linux wireless extension driver interface.

- file driver_ralink.c

  WPA Supplicant - driver interaction with Ralink Wireless Client.

- file driver_ralink.h

  WPA Supplicant - driver_ralink exported functions.

- file driver_roboswitch.c

  WPA Supplicant - roboswitch driver interface.

- file driver_test.c

  WPA Supplicant - testing driver interface.

- file driver_wext.c

  WPA Supplicant - driver interaction with generic Linux Wireless Extensions.

- file driver_wext.h

  WPA Supplicant - driver_wext exported functions.

- file driver_wired.c

  WPA Supplicant - wired Ethernet driver interface.

- file drivers.c

  Driver interface list.

- file **MobileApple80211.c**
- file **MobileApple80211.h**
- file ndis_events.c

  ndis_events - Receive NdisMIndicateStatus() events using WMI

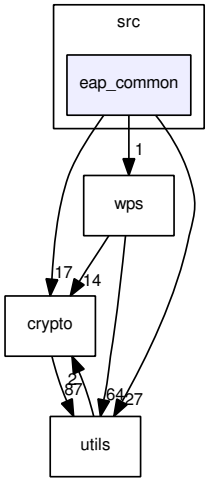
- file **prism54.h**
- file priv_netlink.h

  wpa_supplicant - Private copy of Linux netlink/rtnetlink definitions.

- file scan_helpers.c

  WPA Supplicant - Helper functions for scan result processing.

13.5 src/eap_common/ Directory Reference



Files

- file chap.c

  CHAP-MD5 (RFC 1994).

- file chap.h

  CHAP-MD5 (RFC 1994).

- file eap_common.c

  EAP common peer/server definitions.

- file eap_common.h

  EAP common peer/server definitions.

- file eap_defs.h

  EAP server/peer: Shared EAP definitions.

- file eap_fast_common.c

  EAP-FAST common helper functions (RFC 4851).

- file eap_fast_common.h

  EAP-FAST definitions (RFC 4851).

- file eap_gpsk_common.c

  EAP server/peer: EAP-GPSK shared routines.

- file eap_gpsk_common.h

  EAP server/peer: EAP-GPSK shared routines.

- file eap_ikev2_common.c

EAP-IKEv2 common routines.

- file eap_ikev2_common.h

  EAP-IKEv2 definitions.

- file eap_pax_common.c

  EAP server/peer: EAP-PAX shared routines.

- file eap_pax_common.h

  EAP server/peer: EAP-PAX shared routines.

- file eap_peap_common.c

  EAP-PEAP common routines.

- file eap_peap_common.h

  EAP-PEAP common routines.

- file eap_psk_common.c

  EAP server/peer: EAP-PSK shared routines.

- file eap_psk_common.h

  EAP server/peer: EAP-PSK shared routines.

- file eap_sake_common.c

  EAP server/peer: EAP-SAKE shared routines.

- file eap_sake_common.h

  EAP server/peer: EAP-SAKE shared routines.

- file eap_sim_common.c

  EAP peer/server: EAP-SIM/AKA/AKA' shared routines.

- file eap_sim_common.h

  EAP peer/server: EAP-SIM/AKA/AKA' shared routines.

- file eap_tlv_common.h

  EAP-TLV definitions (draft-josefsson-pppext-eap-tls-eap-10.txt).

- file eap_ttls.h

  EAP server/peer: EAP-TTLS (RFC 5281).

- file eap_wsc_common.c

  EAP-WSC common routines for Wi-Fi Protected Setup.

- file eap_wsc_common.h

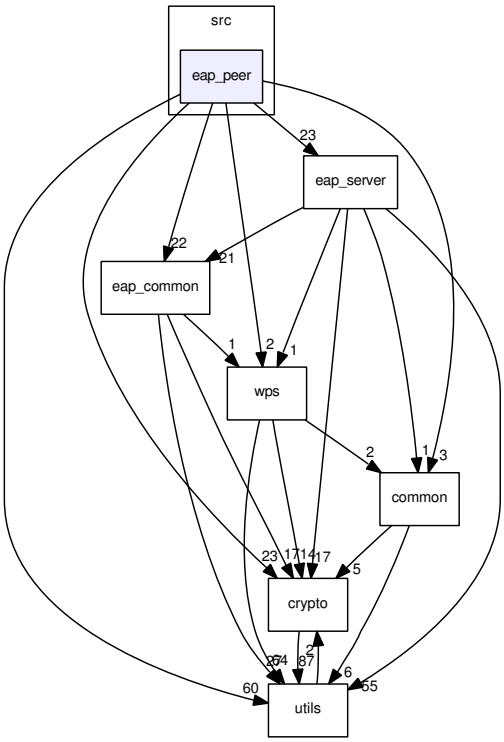
  EAP-WSC definitions for Wi-Fi Protected Setup.

- file ikev2_common.c

  IKEv2 common routines for initiator and responder.

- file ikev2_common.h
  IKEv2 definitions.

13.6 src/eap_peer/ Directory Reference



Files

- file eap.c

  EAP peer state machines (RFC 4137).

- file eap.h

  EAP peer state machine functions (RFC 4137).

- file eap_aka.c

  EAP peer method: EAP-AKA (RFC 4187) and EAP-AKA' (draft-arkko-eap-aka-kdf).

- file eap_config.h

  EAP peer configuration data.

- file eap_fast.c

  EAP peer method: EAP-FAST (RFC 4851).

- file eap_fast_pac.c

  EAP peer method: EAP-FAST PAC file processing.

- file eap_fast_pac.h

  EAP peer method: EAP-FAST PAC file processing.

- file eap_gpsk.c
  EAP peer method: EAP-GPSK (RFC 5433).
- file eap_gtc.c
  EAP peer method: EAP-GTC (RFC 3748).
- file eap_i.h
  EAP peer state machines internal structures (RFC 4137).
- file eap_ikev2.c
  EAP-IKEv2 peer (RFC 5106).
- file eap_leap.c
  EAP peer method: LEAP.
- file eap_md5.c
  EAP peer method: EAP-MD5 (RFC 3748 and RFC 1994).
- file eap_methods.c
  EAP peer: Method registration.
- file eap_methods.h
  EAP peer: Method registration.
- file eap_mschapv2.c
  EAP peer method: EAP-MSCHAPV2 (draft-kamath-pppext-eap-mschapv2-00.txt).
- file eap_otp.c
  EAP peer method: EAP-OTP (RFC 3748).
- file eap_pax.c
  EAP peer method: EAP-PAX (RFC 4746).
- file eap_peap.c
  EAP peer method: EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-10.txt).
- file eap_psk.c
  EAP peer method: EAP-PSK (RFC 4764).
- file eap_sake.c
  EAP peer method: EAP-SAKE (RFC 4763).
- file eap_sim.c
  EAP peer method: EAP-SIM (RFC 4186).
- file eap_tls.c
  EAP peer method: EAP-TLS (RFC 2716).
- file eap_tls_common.c
  EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions.

- file eap_tls_common.h

  EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions.

- file eap_tnc.c

  EAP peer method: EAP-TNC (Trusted Network Connect).

- file eap_ttls.c

  EAP peer method: EAP-TTLS (RFC 5281).

- file eap_vendor_test.c

  EAP peer method: Test method for vendor specific (expanded) EAP type.

- file eap_wsc.c

  EAP-WSC peer for Wi-Fi Protected Setup.

- file ikev2.c

  IKEv2 responder (RFC 4306) for EAP-IKEV2.

- file ikev2.h

  IKEv2 responder (RFC 4306) for EAP-IKEV2.

- file mschapv2.c

  MSCHAPV2 (RFC 2759).

- file mschapv2.h

  MSCHAPV2 (RFC 2759).

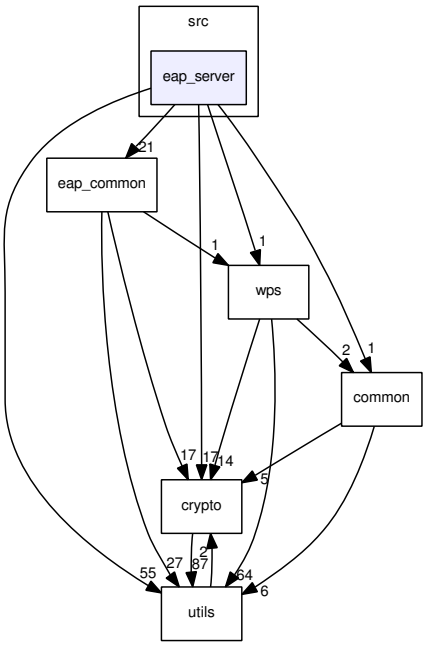
- file tncc.c

  EAP-TNC - TNCC (IF-IMC and IF-TNCCS).

- file tncc.h

  EAP-TNC - TNCC (IF-IMC and IF-TNCCS).

13.7 src/eap_server/ Directory Reference



Files

- file eap.c

  hostapd / EAP Full Authenticator state machine (RFC 4137)

- file eap.h

  hostapd / EAP Full Authenticator state machine (RFC 4137)

- file eap_aka.c

  hostapd / EAP-AKA (RFC 4187) and EAP-AKA' (draft-arkko-eap-aka-kdf)

- file eap_fast.c

  EAP-FAST server (RFC 4851).

- file eap_gpsk.c

  hostapd / EAP-GPSK (RFC 5433) server

- file eap_gtc.c

  hostapd / EAP-GTC (RFC 3748)

- file eap_i.h

  hostapd / EAP Authenticator state machine internal structures (RFC 4137)

- file eap_identity.c

  hostapd / EAP-Identity

- file eap_ikev2.c
  EAP-IKEv2 server (RFC 5106).
- file eap_md5.c
  hostapd / EAP-MD5 server
- file eap_methods.c
  hostapd / EAP method registration
- file eap_methods.h
  hostapd / EAP method registration
- file eap_mschapv2.c
  hostapd / EAP-MSCHAPv2 (draft-kamath-pppext-eap-mschapv2-00.txt) server
- file eap_pax.c
  hostapd / EAP-PAX (RFC 4746) server
- file eap_peap.c
  hostapd / EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-10.txt)
- file eap_psk.c
  hostapd / EAP-PSK (RFC 4764) server
- file eap_sake.c
  hostapd / EAP-SAKE (RFC 4763) server
- file eap_sim.c
  hostapd / EAP-SIM (RFC 4186)
- file eap_sim_db.c
  hostapd / EAP-SIM database/authenticator gateway
- file eap_sim_db.h
  hostapd / EAP-SIM database/authenticator gateway
- file eap_tls.c
  hostapd / EAP-TLS (RFC 2716)
- file eap_tls_common.c
  hostapd / EAP-TLS/PEAP/TTLS/FAST common functions
- file eap_tls_common.h
  hostapd / EAP-TLS/PEAP/TTLS/FAST common functions
- file eap_tnc.c
  EAP server method: EAP-TNC (Trusted Network Connect).
- file eap_ttls.c

hostapd / EAP-TTLS (RFC 5281)

- file eap_vendor_test.c

  hostapd / Test method for vendor specific (expanded) EAP type

- file eap_wsc.c

  EAP-WSC server for Wi-Fi Protected Setup.

- file ikev2.c

  IKEv2 initiator (RFC 4306) for EAP-IKEV2.

- file ikev2.h

  IKEv2 initiator (RFC 4306) for EAP-IKEV2.

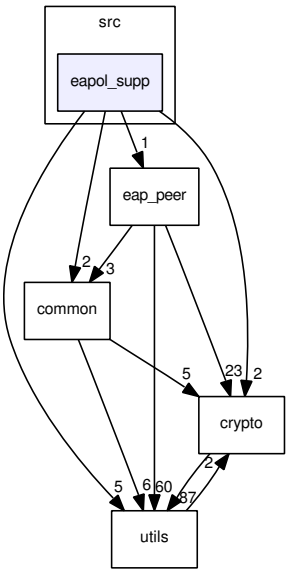
- file tncs.c

  EAP-TNC - TNCS (IF-IMV, IF-TNCCS, and IF-TNCCS-SOH).

- file tncs.h

  EAP-TNC - TNCS (IF-IMV, IF-TNCCS, and IF-TNCCS-SOH).

13.8 src/eapol_supp/ Directory Reference



Files

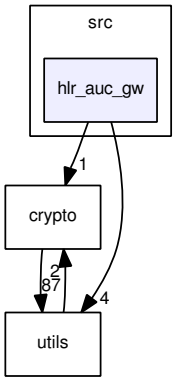
- file eapol_supp_sm.c

  EAPOL supplicant state machines.

- file eapol_supp_sm.h

  EAPOL supplicant state machines.

13.9 src/hlr_auc_gw/ Directory Reference



Files

- file hlr_auc_gw.c

  HLR/AuC testing gateway for hostapd EAP-SIM/AKA database/authenticator.

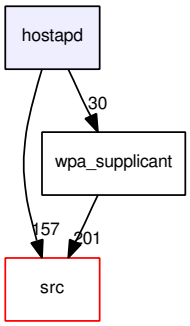
- file milenage.c

  3GPP AKA - Milenage algorithm (3GPP TS 35.205, .206, .207, .208)

- file milenage.h

  UMTS AKA - Milenage algorithm (3GPP TS 35.205, .206, .207, .208).

13.10 hostapd/ Directory Reference



Files

- file accounting.c
  hostapd / RADIUS Accounting
- file accounting.h
  hostapd / RADIUS Accounting
- file ap_list.c
  hostapd / AP table
- file ap_list.h
  hostapd / AP table
- file beacon.c
  hostapd / IEEE 802.11 Management: Beacon and Probe Request/Response
- file beacon.h
  hostapd / IEEE 802.11 Management: Beacon and Probe Request/Response
- file config.c
  hostapd / Configuration file
- file config.h
  hostapd / Configuration file
- file ctrl_iface.c
  hostapd / UNIX domain socket -based control interface
- file ctrl_iface.h
  hostapd / UNIX domain socket -based control interface
- file ctrl_iface_ap.c
  Control interface for shared AP commands.

- file ctrl_iface_ap.h
  Control interface for shared AP commands.
- file driver_i.h
  hostapd - internal driver interface wrappers
- file drv_callbacks.c
  hostapd / Callback functions for driver wrappers
- file eapol_sm.c
  hostapd / IEEE 802.1X-2004 Authenticator - EAPOL state machine
- file eapol_sm.h
  hostapd / IEEE 802.1X-2004 Authenticator - EAPOL state machine
- file hostapd.c
  hostapd / Initialization and configuration
- file hostapd.h
  hostapd / Initialization and configuration Host AP kernel driver
- file hostapd_cli.c
  hostapd - command line interface for hostapd daemon
- file hw_features.c
  hostapd / Hardware feature query and different modes
- file hw_features.h
  hostapd / Hardware feature query and different modes
- file iapp.c
  hostapd / IEEE 802.11F-2003 Inter-Access Point Protocol (IAPP)
- file iapp.h
  hostapd / IEEE 802.11F-2003 Inter-Access Point Protocol (IAPP)
- file ieee802_11.c
  hostapd / IEEE 802.11 Management
- file ieee802_11.h
  hostapd / IEEE 802.11 Management
- file ieee802_11_auth.c
  hostapd / IEEE 802.11 authentication (ACL)
- file ieee802_11_auth.h
  hostapd / IEEE 802.11 authentication (ACL)
- file ieee802_1x.c
  hostapd / IEEE 802.1X-2004 Authenticator

- file ieee802_1x.h
  hostapd / IEEE 802.1X-2004 Authenticator
- file main.c
  hostapd / main()
- file mlme.c
  hostapd / IEEE 802.11 MLME
- file mlme.h
  hostapd / IEEE 802.11 MLME
- file nt_password_hash.c
  hostapd - Plaintext password to NtPasswordHash
- file peerkey.c
  hostapd - PeerKey for Direct Link Setup (DLS)
- file pmksa_cache.c
  hostapd - PMKSA cache for IEEE 802.11i RSN
- file pmksa_cache.h
  hostapd - PMKSA cache for IEEE 802.11i RSN
- file preauth.c
  hostapd - Authenticator for IEEE 802.11i RSN pre-authentication
- file preauth.h
  hostapd - Authenticator for IEEE 802.11i RSN pre-authentication
- file sta_flags.h
  hostapd - driver interface definition
- file sta_info.c
  hostapd / Station table
- file sta_info.h
  hostapd / Station table
- file tkip_countermeasures.c
  hostapd / TKIP countermeasures
- file tkip_countermeasures.h
  hostapd / TKIP countermeasures
- file vlan_init.c
  hostapd / VLAN initialization
- file vlan_init.h

hostapd / VLAN initialization

- file wme.c

  hostapd / WMM (Wi-Fi Multimedia)

- file wme.h

  hostapd / WMM (Wi-Fi Multimedia)

- file wpa.c

  hostapd - IEEE 802.11i-2004 / WPA Authenticator

- file wpa.h

  hostapd - IEEE 802.11i-2004 / WPA Authenticator

- file wpa_auth_i.h

  hostapd - IEEE 802.11i-2004 / WPA Authenticator: Internal definitions

- file wpa_auth_ie.c

  hostapd - WPA/RSN IE and KDE definitions

- file wpa_auth_ie.h

  hostapd - WPA/RSN IE and KDE definitions

- file wpa_ft.c

  hostapd - IEEE 802.11r - Fast BSS Transition

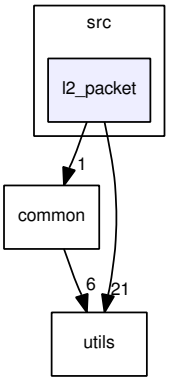
- file wps_hostapd.c

  hostapd / WPS integration

- file wps_hostapd.h

  hostapd / WPS integration

13.11 src/l2_packet/ Directory Reference



Files

- file l2_packet.h

  WPA Supplicant - Layer2 packet interface definition.

- file l2_packet_freebsd.c

  WPA Supplicant - Layer2 packet handling with FreeBSD.

- file l2_packet_linux.c

  WPA Supplicant - Layer2 packet handling with Linux packet sockets.

- file l2_packet_ndis.c

  WPA Supplicant - Layer2 packet handling with Microsoft NDISUIO.

- file l2_packet_none.c

  WPA Supplicant - Layer2 packet handling example with dummy functions.

- file l2_packet_pcap.c

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

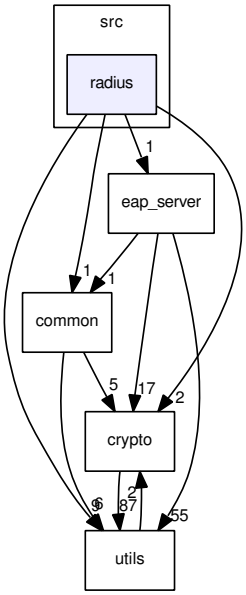
- file l2_packet_privsep.c

  WPA Supplicant - Layer2 packet handling with privilege separation.

- file l2_packet_winpcap.c

  WPA Supplicant - Layer2 packet handling with WinPcap RX thread.

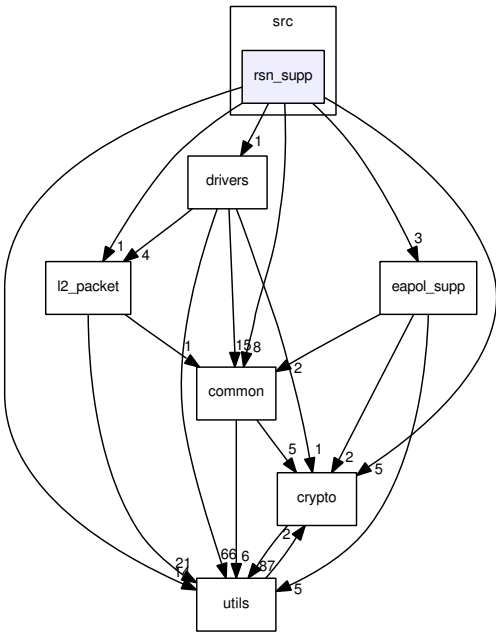
13.12 src/radius/ Directory Reference



Files

- file radius.c
  hostapd / RADIUS message processing
- file radius.h
  hostapd / RADIUS message processing
- file radius_client.c
  RADIUS client.
- file radius_client.h
  RADIUS client.
- file radius_server.c
  hostapd / RADIUS authentication server
- file radius_server.h
  hostapd / RADIUS authentication server

13.13 src/rsn_supp/ Directory Reference



Files

- file peerkey.c

  WPA Supplicant - PeerKey for Direct Link Setup (DLS).

- file peerkey.h

  WPA Supplicant - PeerKey for Direct Link Setup (DLS).

- file pmksa_cache.c

  WPA Supplicant - RSN PMKSA cache.

- file pmksa_cache.h

  wpa_supplicant - WPA2/RSN PMKSA cache functions

- file preauth.c

  WPA Supplicant - RSN pre-authentication.

- file preauth.h

  wpa_supplicant - WPA2/RSN pre-authentication functions

- file wpa.c

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

- file wpa.h

  wpa_supplicant - WPA definitions

- file wpa_ft.c

  WPA Supplicant - IEEE 802.11r - Fast BSS Transition.

- file wpa_i.h

  wpa_supplicant - Internal WPA state machine definitions

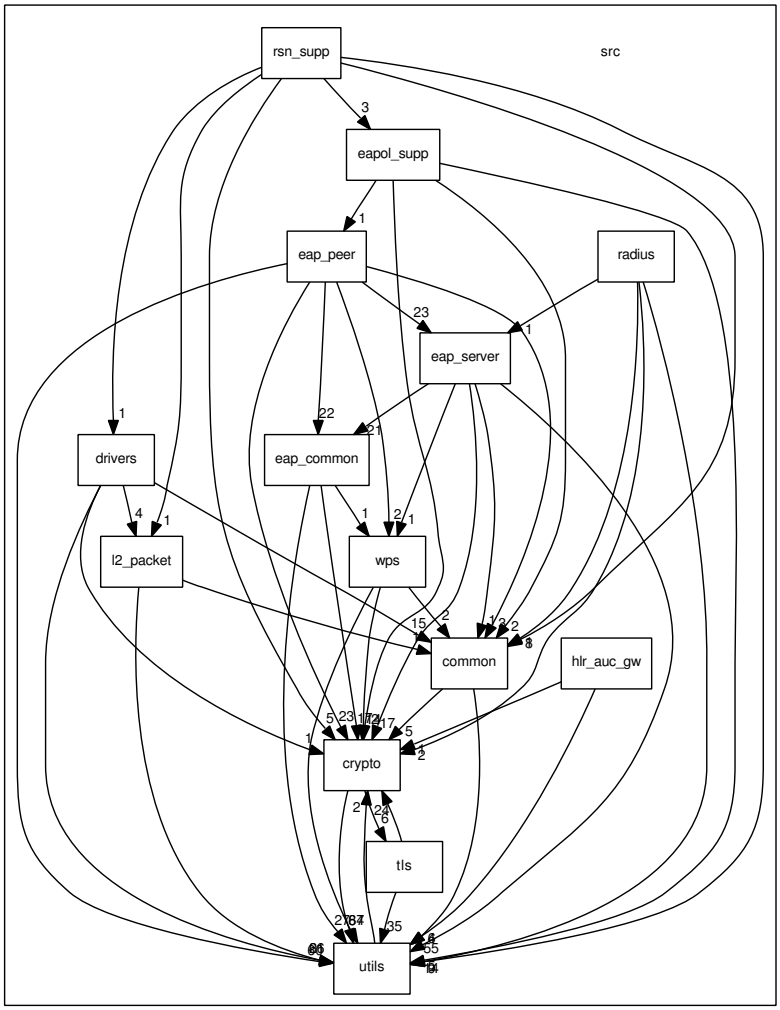
- file wpa_ie.c

  wpa_supplicant - WPA/RSN IE and KDE processing

- file wpa_ie.h

  wpa_supplicant - WPA/RSN IE and KDE definitions

13.14 src/ Directory Reference

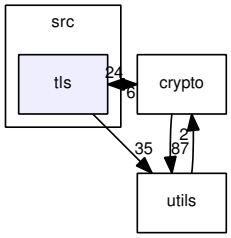


Directories

- directory common
- directory crypto
- directory drivers
- directory eap_common
- directory eap_peer
- directory eap_server
- directory eapol_supp
- directory hlr_auc_gw
- directory l2_packet
- directory radius
- directory rsn_supp
- directory tls
- directory utils

- directory wps

13.15 src/tls/ Directory Reference



Files

- file asn1.c

  ASN.1 DER parsing.

- file asn1.h

  ASN.1 DER parsing.

- file asn1_test.c

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

- file bignum.c

  Big number math.

- file bignum.h

  Big number math.

- file libtommath.c

  Minimal code for RSA support from LibTomMath 0.41 http://libtom.org/ http://libtom.org/files/ltm-0.41.tar.bz2 *This library was released in public domain by Tom St Denis.*

- file pkcs1.c

  PKCS #1 (RSA Encryption).

- file pkcs1.h

  PKCS #1 (RSA Encryption).

- file pkcs5.c

  PKCS #5 (Password-based Encryption).

- file pkcs5.h

  PKCS #5 (Password-based Encryption).

- file pkcs8.c

  PKCS #8 (Private-key information syntax).

- file pkcs8.h
  PKCS #8 (Private-key information syntax).
- file rsa.c
  RSA.
- file rsa.h
  RSA.
- file tlsv1_client.c
  TLSv1 client (RFC 2246).
- file tlsv1_client.h
  TLSv1 client (RFC 2246).
- file tlsv1_client_i.h
  TLSv1 client - internal structures.
- file tlsv1_client_read.c
  TLSv1 client - read handshake message.
- file tlsv1_client_write.c
  TLSv1 client - write handshake message.
- file tlsv1_common.c
  TLSv1 common routines.
- file tlsv1_common.h
  TLSv1 common definitions.
- file tlsv1_cred.c
  TLSv1 credentials.
- file tlsv1_cred.h
  TLSv1 credentials.
- file tlsv1_record.c
  TLSv1 Record Protocol.
- file tlsv1_record.h
  TLSv1 Record Protocol.
- file tlsv1_server.c
  TLSv1 server (RFC 2246).
- file tlsv1_server.h
  TLSv1 server (RFC 2246).
- file tlsv1_server_i.h
  TLSv1 server - internal structures.

- file tlsv1_server_read.c

  TLSv1 server - read handshake message.

- file tlsv1_server_write.c

  TLSv1 server - write handshake message.

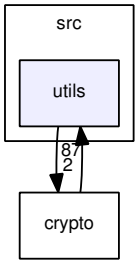
- file x509v3.c

  X.509v3 certificate parsing and processing (RFC 3280 profile).

- file x509v3.h

  X.509v3 certificate parsing and processing.

13.16 src/utils/ Directory Reference



Files

- file base64.c

  Base64 encoding/decoding (RFC1341).

- file base64.h

  Base64 encoding/decoding (RFC1341).

- file build_config.h

  wpa_supplicant/hostapd - Build time configuration defines

- file common.c

  wpa_supplicant/hostapd / common helper functions, etc.

- file common.h

  wpa_supplicant/hostapd / common helper functions, etc.

- file eloop.c

  Event loop based on select() loop.

- file eloop.h

  Event loop.

- file eloop_none.c

  Event loop - empty template (basic structure, but no OS specific operations).

- file eloop_win.c

  Event loop based on Windows events and WaitForMultipleObjects.

- file includes.h

  wpa_supplicant/hostapd - Default include files

- file ip_addr.c

  IP address processing.

- file ip_addr.h

IP address processing.

- file os.h

  wpa_supplicant/hostapd / OS specific functions

- file os_internal.c

  wpa_supplicant/hostapd / Internal implementation of OS specific functions

- file os_none.c

  wpa_supplicant/hostapd / Empty OS specific functions

- file os_unix.c

  wpa_supplicant/hostapd / OS specific functions for UNIX/POSIX systems

- file os_win32.c

  wpa_supplicant/hostapd / OS specific functions for Win32 systems

- file pcsc_funcs.c

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

- file pcsc_funcs.h

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

- file radiotap.c

  Radiotap parser.

- file **radiotap.h**
- file **radiotap_iter.h**
- file state_machine.h

  wpa_supplicant/hostapd - State machine definitions

- file uuid.c

  Universally Unique IDentifier (UUID).

- file uuid.h

  Universally Unique IDentifier (UUID).

- file wpa_debug.c

  wpa_supplicant/hostapd / Debug prints

- file wpa_debug.h

  wpa_supplicant/hostapd / Debug prints

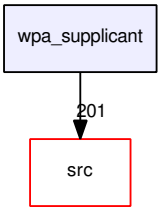
- file wpabuf.c

  Dynamic data buffer.

- file wpabuf.h

  Dynamic data buffer.

13.17 wpa_supplicant/ Directory Reference



Files

- file ap.c
  WPA Supplicant - Basic AP mode support routines.
- file ap.h
  WPA Supplicant - Basic AP mode support routines.
- file bgscan.c
  WPA Supplicant - background scan and roaming interface.
- file bgscan.h
  WPA Supplicant - background scan and roaming interface.
- file bgscan_simple.c
  WPA Supplicant - background scan and roaming module: simple.
- file blacklist.c
  wpa_supplicant - Temporary BSSID blacklist
- file blacklist.h
  wpa_supplicant - Temporary BSSID blacklist
- file config.c
  WPA Supplicant / Configuration parser and common functions.
- file config.h
  WPA Supplicant / Configuration file structures.
- file config_file.c
  WPA Supplicant / Configuration backend: text file.
- file config_none.c
  WPA Supplicant / Configuration backend: empty starting point.
- file config_ssid.h
  WPA Supplicant / Network configuration structures.
- file config_winreg.c

WPA Supplicant / Configuration backend: Windows registry.

- file ctrl_iface.c

  WPA Supplicant / Control interface (shared code for all backends).

- file ctrl_iface.h

  WPA Supplicant / UNIX domain socket -based control interface.

- file ctrl_iface_dbus.c

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_dbus.h

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_dbus_handlers.c

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_dbus_handlers.h

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_dbus_new.c

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_dbus_new.h

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_dbus_new_handlers.c

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_dbus_new_handlers.h

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_dbus_new_helpers.c

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_dbus_new_helpers.h

  WPA Supplicant / dbus-based control interface.

- file ctrl_iface_named_pipe.c

  WPA Supplicant / Windows Named Pipe -based control interface.

- file ctrl_iface_udp.c

  WPA Supplicant / UDP socket -based control interface.

- file ctrl_iface_unix.c

  WPA Supplicant / UNIX domain socket -based control interface.

- file dbus_dict_helpers.c

  WPA Supplicant / dbus-based control interface.

- file dbus_dict_helpers.h
  WPA Supplicant / dbus-based control interface.
- file driver_i.h
  wpa_supplicant - Internal driver interface wrappers
- file eapol_test.c
  WPA Supplicant - test code.
- file events.c
  WPA Supplicant - Driver event processing.
- file ibss_rsn.c
  wpa_supplicant - IBSS RSN
- file ibss_rsn.h
  wpa_supplicant - IBSS RSN
- file main.c
  WPA Supplicant / main() function for UNIX like OSes and MinGW.
- file main_none.c
  WPA Supplicant / Example program entrypoint.
- file main_symbian.cpp
  WPA Supplicant / Program entrypoint for Symbian.
- file main_winmain.c
  WPA Supplicant / WinMain() function for Windows-based applications.
- file main_winsvc.c
  WPA Supplicant / main() function for Win32 service.
- file mlme.c
  WPA Supplicant - Client mode MLME.
- file mlme.h
  WPA Supplicant - Client mode MLME.
- file notify.c
  wpa_supplicant - Event notifications
- file notify.h
  wpa_supplicant - Event notifications
- file preauth_test.c
  WPA Supplicant - test code for pre-authentication.
- file scan.c
  WPA Supplicant - Scanning.

- file sme.c

  wpa_supplicant - SME

- file sme.h

  wpa_supplicant - SME

- file win_if_list.c

  win_if_list - Display network interfaces with description (for Windows)

- file wpa_cli.c

  WPA Supplicant - command line interface for wpa_supplicant daemon.

- file wpa_passphrase.c

  WPA Supplicant - ASCII passphrase to WPA PSK tool.

- file wpa_priv.c

  WPA Supplicant / privileged helper program.

- file wpa_supplicant.c

  WPA Supplicant.

- file wpa_supplicant_i.h

  wpa_supplicant - Internal definitions

- file wpas_glue.c

  WPA Supplicant - Glue code to setup EAPOL and RSN modules.

- file wpas_glue.h

  WPA Supplicant - Glue code to setup EAPOL and RSN modules.

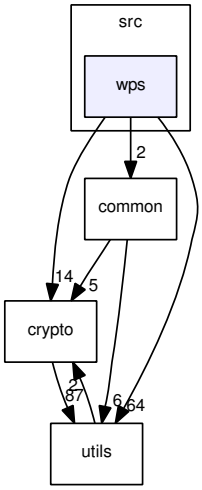
- file wps_supplicant.c

  wpa_supplicant / WPS integration

- file wps_supplicant.h

  wpa_supplicant / WPS integration

13.18 src/wps/ Directory Reference



Files

- file http.h

  HTTP for WPS.

- file http_client.c

  http_client - HTTP client

- file http_client.h

  http_client - HTTP client

- file http_server.c

  http_server - HTTP server

- file http_server.h

  http_server - HTTP server

- file httpread.c

  httpread - Manage reading file(s) from HTTP/TCP socket Author: Ted Merrill

- file httpread.h

  httpread - Manage reading file(s) from HTTP/TCP socket Author: Ted Merrill

- file ndef.c

  NDEF(NFC Data Exchange Format) routines for Wi-Fi Protected Setup Reference is "NFCForum-TS-NDEF_1.0 2006-07-24".

- file upnp_xml.c

  UPnP XML helper routines.

- file upnp_xml.h
  UPnP XML helper routines.
- file wps.c
  Wi-Fi Protected Setup.
- file wps.h
  Wi-Fi Protected Setup.
- file wps_attr_build.c
  Wi-Fi Protected Setup - attribute building.
- file wps_attr_parse.c
  Wi-Fi Protected Setup - attribute parsing.
- file wps_attr_process.c
  Wi-Fi Protected Setup - attribute processing.
- file wps_common.c
  Wi-Fi Protected Setup - common functionality.
- file wps_defs.h
  Wi-Fi Protected Setup - message definitions.
- file wps_dev_attr.c
  Wi-Fi Protected Setup - device attributes.
- file wps_dev_attr.h
  Wi-Fi Protected Setup - device attributes.
- file wps_enrollee.c
  Wi-Fi Protected Setup - Enrollee.
- file wps_er.c
  Wi-Fi Protected Setup - External Registrar.
- file wps_er.h
  Wi-Fi Protected Setup - External Registrar.
- file wps_er_ssdp.c
  Wi-Fi Protected Setup - External Registrar (SSDP).
- file wps_i.h
  Wi-Fi Protected Setup - internal definitions.
- file wps_nfc.c
  NFC routines for Wi-Fi Protected Setup.
- file wps_nfc_pn531.c
  NFC PN531 routines for Wi-Fi Protected Setup.

- file wps_registrar.c

  Wi-Fi Protected Setup - Registrar.

- file wps_ufd.c

  UFD routines for Wi-Fi Protected Setup.

- file wps_upnp.c

  UPnP WPS Device.

- file wps_upnp.h

  UPnP WPS Device.

- file wps_upnp_event.c

  UPnP WPS Device - Event processing.

- file wps_upnp_i.h

  UPnP for WPS / internal definitions.

- file wps_upnp_ssdp.c

  UPnP SSDP for WPS.

- file wps_upnp_web.c

  UPnP WPS Device - Web connections.

Chapter 14

Data Structure Documentation

14.1 _BSSID_INFO Struct Reference

Data Fields

- NDIS_802_11_MAC_ADDRESS **BSSID**
- NDIS_802_11_PMKID_VALUE **PMKID**

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

- src/drivers/driver_ralink.h

14.2 _DOT11_SCAN_REQUEST_V2 Struct Reference

Data Fields

- DOT11_BSS_TYPE **dot11BSSType**
- DOT11_MAC_ADDRESS **dot11BSSID**
- DOT11_SCAN_TYPE **dot11ScanType**
- BOOLEAN **bRestrictedScan**
- ULONG **udot11SSIDsOffset**
- ULONG **uNumOfdot11SSIDs**
- BOOLEAN **bUseRequestIE**
- ULONG **uRequestIDsOffset**
- ULONG **uNumOfRequestIDs**
- ULONG **uPhyTypeInfosOffset**
- ULONG **uNumOfPhyTypeInfos**
- ULONG **uIEsOffset**
- ULONG **uIEsLength**
- UCHAR **ucBuffer** [1]

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

- src/drivers/driver_ndis.c

14.3 _LARGE_INTEGER Union Reference

Data Fields

- struct {
  ULONG **LowPart**
  LONG **HighPart**
  } **vv**

- struct {
  ULONG **LowPart**
  LONG **HighPart**
  } **u**

- s64 **QuadPart**

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

- src/drivers/driver_ralink.h

14.4 _MLME_DEAUTH_REQ_STRUCT Struct Reference

Data Fields

- UCHAR **Addr** [MAC_ADDR_LEN]
- USHORT **Reason**

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

- src/drivers/driver_ralink.h

14.5 _NDIS_802_11_AI_REQFI Struct Reference

Data Fields

- USHORT **Capabilities**
- USHORT **ListenInterval**
- NDIS_802_11_MAC_ADDRESS **CurrentAPAddress**

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

- src/drivers/driver_ralink.h

14.6 _NDIS_802_11_AI_RESFI Struct Reference

Data Fields

- USHORT **Capabilities**
- USHORT **StatusCode**
- USHORT **AssociationId**

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

- src/drivers/driver_ralink.h

14.7 _NDIS_802_11_ASSOCIATION_INFORMATION Struct Reference

Data Fields

- ULONG **Length**
- USHORT **AvailableRequestFixedIEs**
- NDIS_802_11_AI_REQFI **RequestFixedIEs**
- ULONG **RequestIELength**
- ULONG **OffsetRequestIEs**
- USHORT **AvailableResponseFixedIEs**
- NDIS_802_11_AI_RESFI **ResponseFixedIEs**
- ULONG **ResponseIELength**
- ULONG **OffsetResponseIEs**

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

- src/drivers/driver_ralink.h

14.8 _NDIS_802_11_BSSID_LIST Struct Reference

Data Fields

- UINT **NumberOfItems**
- NDIS_WLAN_BSSID **Bssid** [1]

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

- src/drivers/driver_ralink.h

14.9 _NDIS_802_11_BSSID_LIST_EX Struct Reference

Data Fields

- UINT **NumberOfItems**
- NDIS_WLAN_BSSID_EX **Bssid** [1]

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

- src/drivers/driver_ralink.h

14.10 _NDIS_802_11_CONFIGURATION Struct Reference

Data Fields

- ULONG **Length**
- ULONG **BeaconPeriod**
- ULONG **ATIMWindow**
- ULONG **DSConfig**
- NDIS_802_11_CONFIGURATION_FH **FHConfig**

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

- src/drivers/driver_ralink.h

14.11 _NDIS_802_11_CONFIGURATION_FH Struct Reference

Data Fields

- ULONG **Length**
- ULONG **HopPattern**
- ULONG **HopSet**
- ULONG **DwellTime**

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

- src/drivers/driver_ralink.h

14.12 _NDIS_802_11_FIXED_IEs Struct Reference

Data Fields

- UCHAR **Timestamp** [8]
- USHORT **BeaconInterval**
- USHORT **Capabilities**

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

- src/drivers/driver_ralink.h

14.13 _NDIS_802_11_KEY Struct Reference

Data Fields

- UINT **Length**
- UINT **KeyIndex**
- UINT **KeyLength**
- NDIS_802_11_MAC_ADDRESS **BSSID**
- NDIS_802_11_KEY_RSC **KeyRSC**
- UCHAR **KeyMaterial** [1]

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

- src/drivers/driver_ralink.h

14.14 _NDIS_802_11_PMKID Struct Reference

Data Fields

- ULONG **Length**
- ULONG **BSSIDInfoCount**
- BSSID_INFO **BSSIDInfo** [1]

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

- src/drivers/driver_ralink.h

14.15 _NDIS_802_11_PMKID_CANDIDATE_LIST Struct Reference

Data Fields

- ULONG **Version**
- ULONG **NumCandidates**
- PMKID_CANDIDATE **CandidateList** [1]

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

- src/drivers/driver_ralink.h

14.16 _NDIS_802_11_REMOVE_KEY Struct Reference

Data Fields

- UINT **Length**
- UINT **KeyIndex**
- NDIS_802_11_MAC_ADDRESS **BSSID**

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

- src/drivers/driver_ralink.h

14.17 _NDIS_802_11_SSID Struct Reference

Data Fields

- INT **SsidLength**
- UCHAR **Ssid** [NDIS_802_11_LENGTH_SSID]

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

- src/drivers/driver_ralink.h

14.18 _NDIS_802_11_WEP Struct Reference

Data Fields

- UINT **Length**
- UINT **KeyIndex**
- UINT **KeyLength**
- UCHAR **KeyMaterial** [1]

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

- src/drivers/driver_ralink.h

14.19 _NDIS_WLAN_BSSID Struct Reference

Data Fields

- ULONG **Length**
- NDIS_802_11_MAC_ADDRESS **MacAddress**
- UCHAR **Reserved** [2]
- NDIS_802_11_SSID **Ssid**
- ULONG **Privacy**
- NDIS_802_11_RSSI **Rssi**
- NDIS_802_11_NETWORK_TYPE **NetworkTypeInUse**
- NDIS_802_11_CONFIGURATION **Configuration**
- NDIS_802_11_NETWORK_INFRASTRUCTURE **InfrastructureMode**
- NDIS_802_11_RATES **SupportedRates**

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

- src/drivers/driver_ralink.h

14.20 _NDIS_WLAN_BSSID_EX Struct Reference

Data Fields

- ULONG **Length**
- NDIS_802_11_MAC_ADDRESS **MacAddress**
- UCHAR **Reserved** [2]
- NDIS_802_11_SSID **Ssid**
- UINT **Privacy**
- NDIS_802_11_RSSI **Rssi**
- NDIS_802_11_NETWORK_TYPE **NetworkTypeInUse**
- NDIS_802_11_CONFIGURATION **Configuration**
- NDIS_802_11_NETWORK_INFRASTRUCTURE **InfrastructureMode**
- NDIS_802_11_RATES_EX **SupportedRates**
- ULONG **IELength**
- UCHAR **IEs** [1]

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

- src/drivers/driver_ralink.h

14.21 _PMKID_CANDIDATE Struct Reference

Data Fields

- NDIS_802_11_MAC_ADDRESS **BSSID**
- ULONG **Flags**

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

- src/drivers/driver_ralink.h

14.22 _SecPkgContext_EapKeyBlock Struct Reference

Data Fields

- BYTE **rgbKeys** [128]
- BYTE **rgbIVs** [64]

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

- src/crypto/tls_schannel.c

14.23 advertisement_state_machine Struct Reference

Data Fields

- struct advertisement_state_machine * **next**
- struct advertisement_state_machine * **prev**
- struct upnp_wps_device_sm * **sm**
- enum advertisement_type_enum **type**
- int **state**
- int **nerrors**
- struct sockaddr_in **client**

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

- src/wps/wps_upnp_i.h

14.24 ap_driver_data Struct Reference

Data Fields

- struct hostapd_data * **hapd**

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

- wpa_supplicant/ap.c

14.25 ap_info Struct Reference

Data Fields

- struct ap_info * **next**
- struct ap_info * **prev**
- struct ap_info * **hnext**
- struct ap_info * **iter_next**
- struct ap_info * **iter_prev**
- u8 **addr** [6]
- u16 **beacon_int**
- u16 **capability**
- u8 **supported_rates** [WLAN_SUPP_RATES_MAX]
- u8 **ssid** [33]
- size_t **ssid_len**
- int **wpa**
- int **erp**
- int **phytype**
- int **channel**
- int **datarate**
- int **ssi_signal**
- int **ht_support**
- unsigned int **num_beacons**
- time_t **last_beacon**
- int **already_seen**

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

- hostapd/ap_list.h

14.26 asn1_hdr Struct Reference

Data Fields

- const u8 * **payload**
- u8 **identifier**
- u8 **class**
- u8 **constructed**
- unsigned int **tag**
- unsigned int **length**

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

- src/tls/asn1.h

14.27 asn1_oid Struct Reference

Data Fields

- unsigned long **oid** [ASN1_MAX_OID_LEN]
- size_t **len**

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

- src/tls/asn1.h

14.28 wpa_event_data::assoc_info Struct Reference

Data for EVENT_ASSOC and EVENT_ASSOCINFO events.

`#include <driver.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.

14.28.1 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.

14.28.2 Field Documentation

14.28.2.1 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).

14.28.2.2 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).

14.28.2.3 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).

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

- src/drivers/driver.h

14.29 wpa_event_data::assoc_reject Struct Reference

Data for EVENT_ASSOC_REJECT events.

`#include <driver.h>`

Data Fields

- u8 * resp_ies
  (Re)Association Response IEs
- size_t resp_ies_len
  Length of resp_ies in bytes.
- u16 status_code
  Status Code from (Re)association Response.

14.29.1 Detailed Description

Data for EVENT_ASSOC_REJECT events.

14.29.2 Field Documentation

14.29.2.1 u8* wpa_event_data::assoc_reject::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).

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

- src/drivers/driver.h

14.30 atmel_param Struct Reference

Data Fields

- unsigned char **sta_addr** [6]
- int **cmd**
- u8 **alg**
- u8 **key_idx**
- u8 **set_tx**
- u8 **seq** [8]
- u8 **seq_len**
- u16 **key_len**
- u8 **key** [MAX_KEY_LENGTH]
- struct {
  int **reason_code**
  u8 **state**
  } **mlme**

- u8 **pairwise_suite**
- u8 **group_suite**
- u8 **key_mgmt_suite**

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

- src/drivers/driver_atmel.c

14.31 wpa_event_data::auth_info Struct Reference

Data for EVENT_AUTH events.

`#include <driver.h>`

Data Fields

- u8 **peer** [ETH_ALEN]
- u16 **auth_type**
- u16 **status_code**
- const u8 * **ies**
- size_t **ies_len**

14.31.1 Detailed Description

Data for EVENT_AUTH events.

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

- src/drivers/driver.h

14.32 bgscan_ops Struct Reference

Data Fields

- const char * **name**
- void *(* **init**)(struct wpa_supplicant *wpa_s, const char *params, const struct wpa_ssid *ssid)
- void(* **deinit**)(void *priv)
- int(* **notify_scan**)(void *priv)
- void(* **notify_beacon_loss**)(void *priv)
- void(* **notify_signal_change**)(void *priv)

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

- wpa_supplicant/bgscan.h

14.33 bgscan_simple_data Struct Reference

Data Fields

- struct wpa_supplicant * **wpa_s**
- struct wpa_ssid * **ssid**
- int **scan_interval**

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

- wpa_supplicant/bgscan_simple.c

14.34 bss_handler_args Struct Reference

Data Fields

- struct wpa_supplicant * **wpa_s**
- u8 **bssid** [ETH_ALEN]

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

- wpa_supplicant/ctrl_iface_dbus_new_handlers.h

14.35 bss_ie_hdr Struct Reference

Data Fields

- u8 **elem_id**
- u8 **len**
- u8 **oui** [3]

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

- src/drivers/driver_broadcom.c

14.36 BSSID_INFO Struct Reference

Data Fields

- NDIS_802_11_MAC_ADDRESS **BSSID**
- NDIS_802_11_PMKID_VALUE **PMKID**

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

- src/drivers/driver_ndis.c

14.37 cipher_suite_st Struct Reference

Data Fields

- uint8 **suite** [2]

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

- src/crypto/tls_gnutls.c

14.38 tncs_data::conn_imv Struct Reference

Data Fields

- u8 * **imv_send**
- size_t **imv_send_len**
- enum IMV_Action_Recommendation **recommendation**
- int **recommendation_set**

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

- src/eap_server/tncs.c

14.39 crypto_cipher Struct Reference

Data Fields

- gcry_cipher_hd_t **enc**
- gcry_cipher_hd_t **dec**
- EVP_CIPHER_CTX **enc**
- EVP_CIPHER_CTX **dec**

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

- src/crypto/crypto_gnutls.c
- src/crypto/crypto_openssl.c

14.40 crypto_rsa_key Struct Reference

Data Fields

- int **private_key**
- struct bignum * **n**
- struct bignum * **e**
- struct bignum * **d**
- struct bignum * **p**
- struct bignum * **q**
- struct bignum * **dmp1**
- struct bignum * **dmq1**
- struct bignum * **iqmp**

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

- src/tls/rsa.c

14.41 ctrl_iface_dbus_new_priv Struct Reference

Data Fields

- DBusConnection * **con**
- int **should_dispatch**
- void * **application_data**
- u32 **next_objid**

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

- wpa_supplicant/ctrl_iface_dbus_new_helpers.h

14.42 ctrl_iface_dbus_priv Struct Reference

Data Fields

- DBusConnection * **con**
- int **should_dispatch**
- struct wpa_global * **global**
- u32 **next_objid**

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

- wpa_supplicant/ctrl_iface_dbus.c

14.43 ctrl_iface_global_priv Struct Reference

Data Fields

- struct wpa_global * **global**
- struct wpa_global_dst * **ctrl_dst**
- int **sock**
- u8 **cookie** [COOKIE_LEN]

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

- wpa_supplicant/ctrl_iface_named_pipe.c
- wpa_supplicant/ctrl_iface_udp.c
- wpa_supplicant/ctrl_iface_unix.c

14.44 ctrl_iface_priv Struct Reference

Data Fields

- struct wpa_supplicant * **wpa_s**
- struct wpa_ctrl_dst * **ctrl_dst**
- SECURITY_ATTRIBUTES **attr**
- int **sec_attr_set**
- int **sock**
- u8 **cookie** [COOKIE_LEN]

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

- wpa_supplicant/ctrl_iface_named_pipe.c
- wpa_supplicant/ctrl_iface_udp.c
- wpa_supplicant/ctrl_iface_unix.c

14.45 des3_key_s Struct Reference

Data Fields

- u32 **ek** [3][32]
- u32 **dk** [3][32]

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

- src/crypto/des_i.h

14.46 dh_group Struct Reference

Data Fields

- int **id**
- const u8 * **generator**
- size_t **generator_len**
- const u8 * **prime**
- size_t **prime_len**

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

- src/crypto/dh_groups.h

14.47 eap_aka_data Struct Reference

Public Types

- enum {
  CONTINUE, **RESULT_SUCCESS**, **RESULT_FAILURE**, **SUCCESS**,
  FAILURE }
- enum {
  IDENTITY, **CHALLENGE**, **REAUTH**, **NOTIFICATION**,
  SUCCESS, **FAILURE** }

Data Fields

- u8 **ik** [EAP_AKA_IK_LEN]
- u8 **ck** [EAP_AKA_CK_LEN]
- u8 **res** [EAP_AKA_RES_MAX_LEN]
- size_t **res_len**
- u8 **nonce_s** [EAP_SIM_NONCE_S_LEN]
- u8 **mk** [EAP_SIM_MK_LEN]
- u8 **k_aut** [EAP_AKA_PRIME_K_AUT_LEN]
- u8 **k_encr** [EAP_SIM_K_ENCR_LEN]
- u8 **k_re** [EAP_AKA_PRIME_K_RE_LEN]
- u8 **msk** [EAP_SIM_KEYING_DATA_LEN]
- u8 **emsk** [EAP_EMSK_LEN]
- u8 **rand** [EAP_AKA_RAND_LEN]
- u8 **autn** [EAP_AKA_AUTN_LEN]
- u8 **auts** [EAP_AKA_AUTS_LEN]
- int **num_id_req**
- int **num_notification**
- u8 * **pseudonym**
- size_t **pseudonym_len**
- u8 * **reauth_id**
- size_t **reauth_id_len**
- int **reauth**
- unsigned int **counter**
- unsigned int **counter_too_small**
- u8 * **last_eap_identity**
- size_t **last_eap_identity_len**
- enum eap_aka_data:: { ... } **state**
- struct wpabuf * **id_msgs**
- int **prev_id**
- int **result_ind**
- int **use_result_ind**
- u8 **eap_method**
- u8 * **network_name**
- size_t **network_name_len**
- u16 **kdf**
- int **kdf_negotiation**
- enum eap_aka_data:: { ... } **state**

- char * **next_pseudonym**
- char * **next_reauth_id**
- u16 **counter**
- struct eap_sim_reauth * **reauth**
- int **auts_reported**
- u16 **notification**
- int **pending_id**

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

- src/eap_peer/eap_aka.c
- src/eap_server/eap_aka.c

14.48 eap_config Struct Reference

Configuration for EAP state machine.

`#include <eap.h>`

Data Fields

- const char * opensc_engine_path
  OpenSC engine for OpenSSL engine support.
- const char * pkcs11_engine_path
  PKCS#11 engine for OpenSSL engine support.
- const char * pkcs11_module_path
  OpenSC PKCS#11 module for OpenSSL engine.
- struct wps_context * wps
  WPS context data.
- void * **ssl_ctx**
- void * **eap_sim_db_priv**
- Boolean **backend_auth**
- int **eap_server**
- u8 * **pac_opaque_encr_key**
- u8 * **eap_fast_a_id**
- size_t **eap_fast_a_id_len**
- char * **eap_fast_a_id_info**
- int **eap_fast_prov**
- int **pac_key_lifetime**
- int **pac_key_refresh_time**
- int **eap_sim_aka_result_ind**
- int **tnc**
- struct wpabuf * **assoc_wps_ie**
- const u8 * **peer_addr**

14.48.1 Detailed Description

Configuration for EAP state machine.

14.48.2 Field Documentation

14.48.2.1 const char* eap_config::opensc_engine_path

OpenSC engine for OpenSSL engine support. Usually, path to engine_opensc.so.

14.48.2.2 const char* eap_config::pkcs11_engine_path

PKCS#11 engine for OpenSSL engine support. Usually, path to engine_pkcs11.so.

14.48.2.3 const char* eap_config::pkcs11_module_path

OpenSC PKCS#11 module for OpenSSL engine. Usually, path to opensc-pkcs11.so.

14.48.2.4 struct wps_context * eap_config::wps [read]

WPS context data. This is only used by EAP-WSC and can be left NULL if not available.

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

- src/eap_peer/eap.h
- src/eap_server/eap.h

14.49 eap_eapol_interface Struct Reference

Data Fields

- Boolean **eapResp**
- struct wpabuf * **eapRespData**
- Boolean **portEnabled**
- int **retransWhile**
- Boolean **eapRestart**
- int **eapSRTT**
- int **eapRTTVAR**
- Boolean **eapReq**
- Boolean **eapNoReq**
- Boolean **eapSuccess**
- Boolean **eapFail**
- Boolean **eapTimeout**
- struct wpabuf * **eapReqData**
- u8 * **eapKeyData**
- size_t **eapKeyDataLen**
- Boolean **eapKeyAvailable**
- Boolean **aaaEapReq**
- Boolean **aaaEapNoReq**
- Boolean **aaaSuccess**
- Boolean **aaaFail**
- struct wpabuf * **aaaEapReqData**
- u8 * **aaaEapKeyData**
- size_t **aaaEapKeyDataLen**
- Boolean **aaaEapKeyAvailable**
- int **aaaMethodTimeout**
- Boolean **aaaEapResp**
- struct wpabuf * **aaaEapRespData**
- Boolean **aaaTimeout**

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

- src/eap_server/eap.h

14.50 eap_fast_data Struct Reference

Public Types

- enum {
  START, **PHASE1**, **PHASE2_START**, **PHASE2_ID**,
  PHASE2_METHOD, **CRYPTO_BINDING**, **REQUEST_PAC**, **SUCCESS**,
  FAILURE }

Data Fields

- struct eap_ssl_data **ssl**
- int **fast_version**
- struct eap_method * **phase2_method**
- void * **phase2_priv**
- int **phase2_success**
- struct eap_method_type **phase2_type**
- struct eap_method_type * **phase2_types**
- size_t **num_phase2_types**
- int **resuming**
- struct eap_fast_key_block_provisioning * **key_block_p**
- int **provisioning_allowed**
- int **provisioning**
- int **anon_provisioning**
- int **session_ticket_used**
- u8 **key_data** [EAP_FAST_KEY_LEN]
- u8 **emsk** [EAP_EMSK_LEN]
- int **success**
- struct eap_fast_pac * **pac**
- struct eap_fast_pac * **current_pac**
- size_t **max_pac_list_len**
- int **use_pac_binary_format**
- u8 **simck** [EAP_FAST_SIMCK_LEN]
- int **simck_idx**
- struct wpabuf * **pending_phase2_req**
- enum eap_fast_data:: { ... } **state**
- int **force_version**
- int **peer_version**
- u8 **crypto_binding_nonce** [32]
- int **final_result**
- u8 **cmk** [EAP_FAST_CMK_LEN]
- u8 **pac_opaque_encr** [16]
- u8 * **srv_id**
- size_t **srv_id_len**
- char * **srv_id_info**
- int **send_new_pac**
- struct wpabuf * **pending_phase2_resp**
- u8 * **identity**
- size_t **identity_len**

- int **eap_seq**
- int **tnc_started**
- int **pac_key_lifetime**
- int **pac_key_refresh_time**

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

- src/eap_peer/eap_fast.c
- src/eap_server/eap_fast.c

14.51 eap_fast_key_block_provisioning Struct Reference

Data Fields

- u8 **session_key_seed** [EAP_FAST_SKS_LEN]
- u8 **server_challenge** [16]
- u8 **client_challenge** [16]

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

- src/eap_common/eap_fast_common.h

14.52 eap_fast_pac Struct Reference

Data Fields

- struct eap_fast_pac * **next**
- u8 **pac_key** [EAP_FAST_PAC_KEY_LEN]
- u8 * **pac_opaque**
- size_t **pac_opaque_len**
- u8 * **pac_info**
- size_t **pac_info_len**
- u8 * **a_id**
- size_t **a_id_len**
- u8 * **i_id**
- size_t **i_id_len**
- u8 * **a_id_info**
- size_t **a_id_info_len**
- u16 **pac_type**

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

- src/eap_peer/eap_fast_pac.h

14.53 eap_fast_read_ctx Struct Reference

Data Fields

- FILE * **f**
- const char * **pos**
- const char * **end**
- int **line**
- char * **buf**
- size_t **buf_len**

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

- src/eap_peer/eap_fast_pac.c

14.54 eap_fast_tlv_parse Struct Reference

Data Fields

- u8 * **eap_payload_tlv**
- size_t **eap_payload_tlv_len**
- struct eap_tlv_crypto_binding_tlv * **crypto_binding**
- size_t **crypto_binding_len**
- int **iresult**
- int **result**
- int **request_action**
- u8 * **pac**
- size_t **pac_len**

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

- src/eap_common/eap_fast_common.h

14.55 eap_gpsk_csuite Struct Reference

Data Fields

- u8 **vendor** [4]
- u8 **specifier** [2]

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

- src/eap_common/eap_gpsk_common.h

14.56 eap_gpsk_data Struct Reference

Public Types

- enum { **GPSK_1**, **GPSK_3**, **SUCCESS**, **FAILURE** }
- enum { **GPSK_1**, **GPSK_3**, **SUCCESS**, **FAILURE** }

Data Fields

- enum eap_gpsk_data:: { ... } **state**
- u8 **rand_server** [EAP_GPSK_RAND_LEN]
- u8 **rand_peer** [EAP_GPSK_RAND_LEN]
- u8 **msk** [EAP_MSK_LEN]
- u8 **emsk** [EAP_EMSK_LEN]
- u8 **sk** [EAP_GPSK_MAX_SK_LEN]
- size_t **sk_len**
- u8 **pk** [EAP_GPSK_MAX_PK_LEN]
- size_t **pk_len**
- u8 **session_id**
- int **session_id_set**
- u8 * **id_peer**
- size_t **id_peer_len**
- u8 * **id_server**
- size_t **id_server_len**
- int **vendor**
- int **specifier**
- u8 * **psk**
- size_t **psk_len**
- enum eap_gpsk_data:: { ... } **state**
- struct eap_gpsk_csuite **csuite_list** [MAX_NUM_CSUITES]
- size_t **csuite_count**

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

- src/eap_peer/eap_gpsk.c
- src/eap_server/eap_gpsk.c

14.57 eap_gtc_data Struct Reference

Public Types

- enum { **CONTINUE**, **SUCCESS**, **FAILURE** }

Data Fields

- int **prefix**
- enum eap_gtc_data:: { ... } **state**

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

- src/eap_peer/eap_gtc.c
- src/eap_server/eap_gtc.c

14.58 eap_hdr Struct Reference

Data Fields

- u8 **code**
- u8 **identifier**
- be16 **length**

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

- src/eap_common/eap_defs.h

14.59 eap_identity_data Struct Reference

Public Types

- enum { **CONTINUE**, **SUCCESS**, **FAILURE** }

Data Fields

- enum eap_identity_data:: { ... } **state**
- int **pick_up**

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

- src/eap_server/eap_identity.c

14.60 eap_ikev2_data Struct Reference

Public Types

- enum {
  WAIT_START, **PROC_MSG**, **WAIT_FRAG_ACK**, **DONE**,
  FAIL }
- enum {
  MSG, **FRAG_ACK**, **WAIT_FRAG_ACK**, **DONE**,
  FAIL }

Data Fields

- struct ikev2_responder_data **ikev2**
- enum eap_ikev2_data:: { ... } **state**
- struct wpabuf * **in_buf**
- struct wpabuf * **out_buf**
- size_t **out_used**
- size_t **fragment_size**
- int **keys_ready**
- u8 **keymat** [EAP_MSK_LEN+EAP_EMSK_LEN]
- int **keymat_ok**
- struct ikev2_initiator_data **ikev2**
- enum eap_ikev2_data:: { ... } **state**

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

- src/eap_peer/eap_ikev2.c
- src/eap_server/eap_ikev2.c

14.61 eap_key_data Struct Reference

Data Fields

- u8 **encr_key** [IEEE8021X_ENCR_KEY_LEN]
- u8 **sign_key** [IEEE8021X_SIGN_KEY_LEN]

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

- src/eapol_supp/eapol_supp_sm.c

14.62 eap_leap_data Struct Reference

Public Types

- enum { **LEAP_WAIT_CHALLENGE**, **LEAP_WAIT_SUCCESS**, **LEAP_WAIT_RESPONSE**, **LEAP_DONE** }

Data Fields

- enum eap_leap_data:: { ... } **state**
- u8 **peer_challenge** [LEAP_CHALLENGE_LEN]
- u8 **peer_response** [LEAP_RESPONSE_LEN]
- u8 **ap_challenge** [LEAP_CHALLENGE_LEN]
- u8 **ap_response** [LEAP_RESPONSE_LEN]

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

- src/eap_peer/eap_leap.c

14.63 eap_md5_data Struct Reference

Public Types

- enum { **CONTINUE**, **SUCCESS**, **FAILURE** }

Data Fields

- u8 **challenge** [CHALLENGE_LEN]
- enum eap_md5_data:: { ... } **state**

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

- src/eap_server/eap_md5.c

14.64 eap_method Struct Reference

EAP method interface.

`#include <eap_i.h>`

Data Fields

- int vendor
  EAP Vendor-ID (EAP_VENDOR_*) (0 = IETF).
- EapType method
  EAP type number (EAP_TYPE_*).
- const char * name
  Name of the method (e.g., "TLS").
- void *(* init)(struct eap_sm *sm)
  Initialize an EAP method.
- void(* deinit)(struct eap_sm *sm, void *priv)
  Deinitialize an EAP method.
- struct wpabuf *(* process)(struct eap_sm *sm, void *priv, struct eap_method_ret *ret, const struct wpabuf *reqData)
  Process an EAP request.
- Boolean(* isKeyAvailable)(struct eap_sm *sm, void *priv)
  Find out whether EAP method has keying material.
- u8 *(* getKey)(struct eap_sm *sm, void *priv, size_t *len)
  Get EAP method specific keying material (eapKeyData).
- int(* get_status)(struct eap_sm *sm, void *priv, char *buf, size_t buflen, int verbose)
  Get EAP method status.
- Boolean(* has_reauth_data)(struct eap_sm *sm, void *priv)
  Whether method is ready for fast reauthentication.
- void(* deinit_for_reauth)(struct eap_sm *sm, void *priv)
  Release data that is not needed for fast re-auth.
- void *(* init_for_reauth)(struct eap_sm *sm, void *priv)
  Prepare for start of fast re-authentication.
- const u8 *(* get_identity)(struct eap_sm *sm, void *priv, size_t *len)
  Get method specific identity for re-authentication.
- void(* free)(struct eap_method *method)
  Free EAP method data.

- int version
  Version of the EAP peer method interface.
- struct eap_method * next
  Pointer to the next EAP method.
- u8 *(* get_emsk)(struct eap_sm *sm, void *priv, size_t *len)
  Get EAP method specific keying extended material (EMSK).
- void *(* **initPickUp**)(struct eap_sm *sm)
- void(* **reset**)(struct eap_sm *sm, void *priv)
- struct wpabuf *(* **buildReq**)(struct eap_sm *sm, void *priv, u8 id)
- int(* **getTimeout**)(struct eap_sm *sm, void *priv)
- Boolean(* **check**)(struct eap_sm *sm, void *priv, struct wpabuf *respData)
- void(* **process**)(struct eap_sm *sm, void *priv, struct wpabuf *respData)
- Boolean(* **isDone**)(struct eap_sm *sm, void *priv)
- Boolean(* **isSuccess**)(struct eap_sm *sm, void *priv)

14.64.1 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.

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 5.4 of RFC 4137.

14.64.2 Field Documentation

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

Deinitialize an EAP method.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

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

Deinitialize the EAP method and free any allocated private data.

14.64.2.2 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_peer_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

14.64.2.3 void(* eap_method::free)(struct eap_method *method) (struct eap_method * *method*)

Free EAP method data.

Parameters:

method Pointer to the method data registered with eap_peer_method_register().

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

Parameters:

method Pointer to the method data registered with eap_server_method_register().

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

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

Get EAP method specific keying extended material (EMSK).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

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

len Pointer to a variable to store EMSK length

Returns:

EMSK or NULL if not available

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

Parameters:

sm Pointer to EAP state machine allocated with eap_sm_init()

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

len Pointer to a variable to store EMSK length

Returns:

EMSK or NULL if not available

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

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

Get method specific identity for re-authentication.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

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

Get EAP method status.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

14.64.2.7 u8 *(* eap_method::getKey)(struct eap_sm *sm, void *priv, size_t *len) (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_peer_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.

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

Whether method is ready for fast reauthentication.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

14.64.2.9 void ∗(∗ eap_method::init)(struct eap_sm ∗sm) (struct eap_sm ∗ *sm*)

Initialize an EAP method.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Returns:

Pointer to allocated private data, or NULL on failure

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

14.64.2.10 void∗(∗ eap_method::init_for_reauth)(struct eap_sm ∗sm, void ∗priv)

Prepare for start of fast re-authentication.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

14.64.2.11 Boolean(∗ eap_method::isKeyAvailable)(struct eap_sm ∗sm, void ∗priv)

Find out whether EAP method has keying material.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

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

Returns:

TRUE if key material (eapKeyData) is available

14.64.2.12 struct eap_method * eap_method::next [read]

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

14.64.2.13 struct wpabuf*(* eap_method::process)(struct eap_sm *sm, void *priv, struct eap_method_ret *ret, const struct wpabuf *reqData) [read]

Process an EAP request.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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)

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.

14.64.2.14 int eap_method::version

Version of the EAP peer method interface. Version of the EAP server method interface.

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

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

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

- src/eap_peer/eap_i.h
- src/eap_server/eap_i.h

14.65 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).

14.65.1 Detailed Description

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

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

- src/eap_peer/eap_i.h

14.66 eap_method_type Struct Reference

Data Fields

- int **vendor**
- u32 **method**

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

- src/eap_peer/eap.h

14.67 eap_mschapv2_data Struct Reference

Public Types

- enum {
  CHALLENGE, **SUCCESS_REQ**, **FAILURE_REQ**, **SUCCESS**,
  FAILURE }

Data Fields

- u8 **auth_response** [MSCHAPV2_AUTH_RESPONSE_LEN]
- int **auth_response_valid**
- int **prev_error**
- u8 **passwd_change_challenge** [PASSWD_CHANGE_CHAL_LEN]
- int **passwd_change_challenge_valid**
- int **passwd_change_version**
- u8 * **peer_challenge**
- u8 * **auth_challenge**
- int **phase2**
- u8 **master_key** [MSCHAPV2_MASTER_KEY_LEN]
- int **master_key_valid**
- int **success**
- struct wpabuf * **prev_challenge**
- u8 **auth_challenge** [CHALLENGE_LEN]
- int **auth_challenge_from_tls**
- enum eap_mschapv2_data:: { ... } **state**
- u8 **resp_mschapv2_id**

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

- src/eap_peer/eap_mschapv2.c
- src/eap_server/eap_mschapv2.c

14.68 eap_mschapv2_hdr Struct Reference

Data Fields

- u8 **op_code**
- u8 **mschapv2_id**
- u8 **ms_length** [2]

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

- src/eap_peer/eap_mschapv2.c
- src/eap_server/eap_mschapv2.c

14.69 eap_pax_data Struct Reference

Public Types

- enum { **PAX_INIT**, **PAX_STD_2_SENT**, **PAX_DONE** }
- enum { **PAX_STD_1**, **PAX_STD_3**, **SUCCESS**, **FAILURE** }

Data Fields

- enum eap_pax_data:: { ... } **state**
- u8 **mac_id**
- u8 **dh_group_id**
- u8 **public_key_id**
- union {
  u8 **e** [2 ∗EAP_PAX_RAND_LEN]
  struct {
  u8 **x** [EAP_PAX_RAND_LEN]
  u8 **y** [EAP_PAX_RAND_LEN]
  } **r**
  } **rand**

- char ∗ **cid**
- size_t **cid_len**
- u8 **ak** [EAP_PAX_AK_LEN]
- u8 **mk** [EAP_PAX_MK_LEN]
- u8 **ck** [EAP_PAX_CK_LEN]
- u8 **ick** [EAP_PAX_ICK_LEN]
- enum eap_pax_data:: { ... } **state**
- union {
  u8 **e** [2 ∗EAP_PAX_RAND_LEN]
  struct {
  u8 **x** [EAP_PAX_RAND_LEN]
  u8 **y** [EAP_PAX_RAND_LEN]
  } **r**
  } **rand**

- int **keys_set**

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

- src/eap_peer/eap_pax.c
- src/eap_server/eap_pax.c

14.70 eap_pax_hdr Struct Reference

Data Fields

- u8 **op_code**
- u8 **flags**
- u8 **mac_id**
- u8 **dh_group_id**
- u8 **public_key_id**

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

- src/eap_common/eap_pax_common.h

14.71 eap_peap_data Struct Reference

Public Types

- enum { **NO_BINDING**, **OPTIONAL_BINDING**, **REQUIRE_BINDING** }
- enum {
  START, **PHASE1**, **PHASE1_ID2**, **PHASE2_START**,
  PHASE2_ID, **PHASE2_METHOD**, **PHASE2_SOH**, **PHASE2_TLV**,
  SUCCESS_REQ, **FAILURE_REQ**, **SUCCESS**, **FAILURE** }
- enum { **TLV_REQ_NONE**, **TLV_REQ_SUCCESS**, **TLV_REQ_FAILURE** }
- enum { **NO_BINDING**, **OPTIONAL_BINDING**, **REQUIRE_BINDING** }

Data Fields

- struct eap_ssl_data **ssl**
- int **peap_version**
- int **force_peap_version**
- int **force_new_label**
- struct eap_method * **phase2_method**
- void * **phase2_priv**
- int **phase2_success**
- int **phase2_eap_success**
- int **phase2_eap_started**
- struct eap_method_type **phase2_type**
- struct eap_method_type * **phase2_types**
- size_t **num_phase2_types**
- int **peap_outer_success**
- int **resuming**
- int **reauth**
- u8 * **key_data**
- struct wpabuf * **pending_phase2_req**
- enum eap_peap_data:: { ... } **crypto_binding**
- int **crypto_binding_used**
- u8 **binding_nonce** [32]
- u8 **ipmk** [40]
- u8 **cmk** [20]
- int **soh**
- enum eap_peap_data:: { ... } **state**
- int **recv_version**
- int **force_version**
- struct wpabuf * **pending_phase2_resp**
- enum eap_peap_data:: { ... } **tlv_request**
- int **crypto_binding_sent**
- enum eap_peap_data:: { ... } **crypto_binding**
- u8 * **phase2_key**
- size_t **phase2_key_len**
- struct wpabuf * **soh_response**

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

- src/eap_peer/eap_peap.c
- src/eap_server/eap_peap.c

14.72 eap_peer_config Struct Reference

EAP peer configuration/credentials.

`#include <eap_config.h>`

Data Fields

- 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 * 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.
- struct eap_method_type * eap_methods
  Allowed EAP methods.
- char * phase1
  Phase 1 (outer authentication) parameters.
- char * phase2
  Phase2 (inner authentication with TLS tunnel) parameters.
- char * pcsc
  Parameters for PC/SC smartcard interface for USIM and GSM SIM.
- char * pin
  PIN for USIM, GSM SIM, and smartcards.
- int engine
  Enable OpenSSL engine (e.g., for smartcard access).
- char * engine_id
  Engine ID for OpenSSL engine.
- int engine2
  Enable OpenSSL engine (e.g., for smartcard) (Phase 2).
- char * pin2
  PIN for USIM, GSM SIM, and smartcards (Phase 2).

- char * engine2_id
  Engine ID for OpenSSL engine (Phase 2).
- char * key_id
  Key ID for OpenSSL engine.
- char * cert_id
  Cert ID for OpenSSL engine.
- char * ca_cert_id
  CA Cert ID for OpenSSL engine.
- char * key2_id
  Key ID for OpenSSL engine (phase2).
- char * cert2_id
  Cert ID for OpenSSL engine (phase2).
- char * ca_cert2_id
  CA Cert ID for OpenSSL engine (phase2).
- 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.
- char * pac_file

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

- 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 fragment_size

  Maximum EAP fragment size in bytes (default 1398).

- u32 flags

  Network configuration flags (bitfield).

14.72.1 Detailed Description

EAP peer configuration/credentials.

14.72.2 Field Documentation

14.72.2.1 u8* eap_peer_config::altsubject_match

Constraint for server certificate alt. subject. Semicolon separated string of entries to be matched against the alternative subject name of the authentication server certificate. If this string is set, the server sertificate is only accepted if it contains one of the entries in an alternative subject name extension.

altSubjectName string is in following format: TYPE:VALUE

Example: EMAIL:server@example.com Example: DNS:server.example.com;DNS:server2.example.com

Following types are supported: EMAIL, DNS, URI

14.72.2.2 u8* eap_peer_config::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.

14.72.2.3 u8* eap_peer_config::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.

If not set, the identity field will be used for both unencrypted and protected fields.

14.72.2.4 u8* eap_peer_config::ca_cert

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

Alternatively, a named configuration blob can be used by setting this to blob://blob_name.

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

14.72.2.5 u8* eap_peer_config::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.

14.72.2.6 char* eap_peer_config::ca_cert2_id

CA Cert ID for OpenSSL engine (phase2). This is used if the CA certificate for EAP-TLS is on a smartcard.

14.72.2.7 char* eap_peer_config::ca_cert_id

CA Cert ID for OpenSSL engine. This is used if the CA certificate for EAP-TLS is on a smartcard.

14.72.2.8 u8* eap_peer_config::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.

14.72.2.9 u8* eap_peer_config::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.

14.72.2.10 char* eap_peer_config::cert2_id

Cert ID for OpenSSL engine (phase2). This is used if the certificate operations for EAP-TLS are performed using a smartcard.

14.72.2.11 char* eap_peer_config::cert_id

Cert ID for OpenSSL engine. This is used if the certificate operations for EAP-TLS are performed using a smartcard.

14.72.2.12 u8* eap_peer_config::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.

14.72.2.13 u8* eap_peer_config::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.

14.72.2.14 u8* eap_peer_config::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.

14.72.2.15 u8* eap_peer_config::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.

14.72.2.16 struct eap_method_type* eap_peer_config::eap_methods [read]

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

14.72.2.17 int eap_peer_config::engine

Enable OpenSSL engine (e.g., for smartcard access). This is used if private key operations for EAP-TLS are performed using a smartcard.

14.72.2.18 int eap_peer_config::engine2

Enable OpenSSL engine (e.g., for smartcard) (Phase 2). This is used if private key operations for EAP-TLS are performed using a smartcard.

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

14.72.2.19 char* eap_peer_config::engine2_id

Engine ID for OpenSSL engine (Phase 2). "opensc" to select OpenSC engine or "pkcs11" to select PKCS#11 engine.

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

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

14.72.2.20 char* eap_peer_config::engine_id

Engine ID for OpenSSL engine. "opensc" to select OpenSC engine or "pkcs11" to select PKCS#11 engine.

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

14.72.2.21 u32 eap_peer_config::flags

Network configuration flags (bitfield). This variable is used for internal flags to describe further details for the network parameters. bit 0 = password is represented as a 16-byte NtPasswordHash value instead of plaintext password

14.72.2.22 int eap_peer_config::fragment_size

Maximum EAP fragment size in bytes (default 1398). This value limits the fragment size for EAP methods that support fragmentation (e.g., EAP-TLS and EAP-PEAP). This value should be set small enough to make the EAP messages fit in MTU of the network interface used for EAPOL. The default value is suitable for most cases.

14.72.2.23 u8* eap_peer_config::identity

EAP Identity. This field is used to set the real user identity or NAI (for EAP-PSK/PAX/SAKE/GPSK).

14.72.2.24 char* eap_peer_config::key2_id

Key ID for OpenSSL engine (phase2). This is used if private key operations for EAP-TLS are performed using a smartcard.

14.72.2.25 char* eap_peer_config::key_id

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

14.72.2.26 int eap_peer_config::mschapv2_retry

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

14.72.2.27 u8* eap_peer_config::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.

14.72.2.28 u8* eap_peer_config::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.

14.72.2.29 char* eap_peer_config::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.

14.72.2.30 u8* eap_peer_config::password

Password string for EAP. This field can include either the plaintext password (default option) or a NtPasswordHash (16-byte MD4 hash of the unicode presentation of the password) if flags field has EAP_CONFIG_FLAGS_PASSWORD_NTHASH bit set to 1. NtPasswordHash can only be used with authentication mechanism that use this hash as the starting point for operation: MSCHAP and MSCHAPv2 (EAP-MSCHAPv2, EAP-TTLS/MSCHAPv2, EAP-TTLS/MSCHAP, LEAP).

In addition, this field is used to configure a pre-shared key for EAP-PSK/PAX/SAKE/GPSK. The length of the PSK must be 16 for EAP-PSK and EAP-PAX and 32 for EAP-SAKE. EAP-GPSK can use a variable length PSK.

14.72.2.31 char* eap_peer_config::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.

14.72.2.32 int eap_peer_config::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.

14.72.2.33 int eap_peer_config::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.

14.72.2.34 char* eap_peer_config::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.

14.72.2.35 int eap_peer_config::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.

14.72.2.36 int eap_peer_config::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.

14.72.2.37 int eap_peer_config::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.

14.72.2.38 char* eap_peer_config::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).

result_ind=1 can be used to enable EAP-SIM and EAP-AKA to use protected result indication.

fast_provisioning option can be used to enable in-line provisioning of EAP-FAST credentials (PAC): 0 = disabled, 1 = allow unauthenticated provisioning, 2 = allow authenticated provisioning, 3 = allow both unauthenticated and authenticated provisioning

fast_max_pac_list_len=num option can be used to set the maximum number of PAC entries to store in a PAC list (default: 10).

fast_pac_format=binary option can be used to select binary format for storing PAC entries in order to save some space (the default text format uses about 2.5 times the size of minimal binary format).

crypto_binding option can be used to control PEAPv0 cryptobinding behavior: 0 = do not use cryptobinding (default) 1 = use cryptobinding if server supports it 2 = require cryptobinding

EAP-WSC (WPS) uses following options: pin=Device_Password and uuid=Device_UUID

14.72.2.39 char* eap_peer_config::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.

14.72.2.40 char* eap_peer_config::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.

14.72.2.41 char* eap_peer_config::pin2

PIN for USIM, GSM SIM, and smartcards (Phase 2). 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.

This field is like pin2, but used for phase 2 (inside EAP-TTLS/PEAP/FAST tunnel) authentication.

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

14.72.2.42 u8* eap_peer_config::private_key

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

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

cert://substring_to_match

hash://certificate_thumbprint_in_hex

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

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

Alternatively, a named configuration blob can be used by setting this to blob://blob_name.

14.72.2.43 u8* eap_peer_config::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.

14.72.2.44 u8* eap_peer_config::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.

14.72.2.45 u8* eap_peer_config::private_key_passwd

Password for private key file. If left out, this will be asked through control interface.

14.72.2.46 u8* eap_peer_config::subject_match

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

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

.example.com

14.72.2.47 u8* eap_peer_config::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.

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

- src/eap_peer/eap_config.h

14.73 eap_psk_data Struct Reference

Public Types

- enum { **PSK_INIT**, **PSK_MAC_SENT**, **PSK_DONE** }
- enum { **PSK_1**, **PSK_3**, **SUCCESS**, **FAILURE** }

Data Fields

- enum eap_psk_data:: { ... } **state**
- u8 **rand_p** [EAP_PSK_RAND_LEN]
- u8 **ak** [EAP_PSK_AK_LEN]
- u8 **kdk** [EAP_PSK_KDK_LEN]
- u8 **tek** [EAP_PSK_TEK_LEN]
- u8 * **id_s**
- u8 * **id_p**
- size_t **id_s_len**
- size_t **id_p_len**
- u8 **msk** [EAP_MSK_LEN]
- u8 **emsk** [EAP_EMSK_LEN]
- enum eap_psk_data:: { ... } **state**
- u8 **rand_s** [EAP_PSK_RAND_LEN]

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

- src/eap_peer/eap_psk.c
- src/eap_server/eap_psk.c

14.74 eap_psk_hdr_1 Struct Reference

Data Fields

- u8 **flags**
- u8 **rand_s** [EAP_PSK_RAND_LEN]

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

- src/eap_common/eap_psk_common.h

14.75 eap_psk_hdr_2 Struct Reference

Data Fields

- u8 **flags**
- u8 **rand_s** [EAP_PSK_RAND_LEN]
- u8 **rand_p** [EAP_PSK_RAND_LEN]
- u8 **mac_p** [EAP_PSK_MAC_LEN]

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

- src/eap_common/eap_psk_common.h

14.76 eap_psk_hdr_3 Struct Reference

Data Fields

- u8 **flags**
- u8 **rand_s** [EAP_PSK_RAND_LEN]
- u8 **mac_s** [EAP_PSK_MAC_LEN]

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

- src/eap_common/eap_psk_common.h

14.77 eap_psk_hdr_4 Struct Reference

Data Fields

- u8 **flags**
- u8 **rand_s** [EAP_PSK_RAND_LEN]

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

- src/eap_common/eap_psk_common.h

14.78 eap_sake_data Struct Reference

Public Types

- enum {
  IDENTITY, **CHALLENGE**, **CONFIRM**, **SUCCESS**,
  FAILURE }
- enum {
  IDENTITY, **CHALLENGE**, **CONFIRM**, **SUCCESS**,
  FAILURE }

Data Fields

- enum eap_sake_data:: { ... } **state**
- u8 **root_secret_a** [EAP_SAKE_ROOT_SECRET_LEN]
- u8 **root_secret_b** [EAP_SAKE_ROOT_SECRET_LEN]
- u8 **rand_s** [EAP_SAKE_RAND_LEN]
- u8 **rand_p** [EAP_SAKE_RAND_LEN]
- struct {
  u8 **auth** [EAP_SAKE_TEK_AUTH_LEN]
  u8 **cipher** [EAP_SAKE_TEK_CIPHER_LEN]
  } **tek**

- u8 **msk** [EAP_MSK_LEN]
- u8 **emsk** [EAP_EMSK_LEN]
- u8 **session_id**
- int **session_id_set**
- u8 * **peerid**
- size_t **peerid_len**
- u8 * **serverid**
- size_t **serverid_len**
- enum eap_sake_data:: { ... } **state**
- struct {
  u8 **auth** [EAP_SAKE_TEK_AUTH_LEN]
  u8 **cipher** [EAP_SAKE_TEK_CIPHER_LEN]
  } **tek**

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

- src/eap_peer/eap_sake.c
- src/eap_server/eap_sake.c

14.79 eap_sake_hdr Struct Reference

Data Fields

- u8 **version**
- u8 **session_id**
- u8 **subtype**

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

- src/eap_common/eap_sake_common.h

14.80 eap_sake_parse_attr Struct Reference

Data Fields

- const u8 * **rand_s**
- const u8 * **rand_p**
- const u8 * **mic_s**
- const u8 * **mic_p**
- const u8 * **serverid**
- size_t **serverid_len**
- const u8 * **peerid**
- size_t **peerid_len**
- const u8 * **spi_s**
- size_t **spi_s_len**
- const u8 * **spi_p**
- size_t **spi_p_len**
- const u8 * **any_id_req**
- const u8 * **perm_id_req**
- const u8 * **encr_data**
- size_t **encr_data_len**
- const u8 * **iv**
- size_t **iv_len**
- const u8 * **next_tmpid**
- size_t **next_tmpid_len**
- const u8 * **msk_life**

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

- src/eap_common/eap_sake_common.h

14.81 eap_sim_attrs Struct Reference

Data Fields

- const u8 * **rand**
- const u8 * **autn**
- const u8 * **mac**
- const u8 * **iv**
- const u8 * **encr_data**
- const u8 * **version_list**
- const u8 * **nonce_s**
- const u8 * **next_pseudonym**
- const u8 * **next_reauth_id**
- const u8 * **nonce_mt**
- const u8 * **identity**
- const u8 * **res**
- const u8 * **auts**
- const u8 * **checkcode**
- const u8 * **kdf_input**
- const u8 * **bidding**
- size_t **num_chal**
- size_t **version_list_len**
- size_t **encr_data_len**
- size_t **next_pseudonym_len**
- size_t **next_reauth_id_len**
- size_t **identity_len**
- size_t **res_len**
- size_t **res_len_bits**
- size_t **checkcode_len**
- size_t **kdf_input_len**
- enum eap_sim_id_req **id_req**
- int **notification**
- int **counter**
- int **selected_version**
- int **client_error_code**
- int **counter_too_small**
- int **result_ind**
- u16 **kdf** [EAP_AKA_PRIME_KDF_MAX]
- size_t **kdf_count**

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

- src/eap_common/eap_sim_common.h

14.82 eap_sim_data Struct Reference

Public Types

- enum {
  CONTINUE, **RESULT_SUCCESS**, **RESULT_FAILURE**, **SUCCESS**,
  FAILURE }
- enum {
  START, **CHALLENGE**, **REAUTH**, **NOTIFICATION**,
  SUCCESS, **FAILURE** }

Data Fields

- u8 ∗ **ver_list**
- size_t **ver_list_len**
- int **selected_version**
- size_t **min_num_chal**
- size_t **num_chal**
- u8 **kc** [3][EAP_SIM_KC_LEN]
- u8 **sres** [3][EAP_SIM_SRES_LEN]
- u8 **nonce_mt** [EAP_SIM_NONCE_MT_LEN]
- u8 **nonce_s** [EAP_SIM_NONCE_S_LEN]
- u8 **mk** [EAP_SIM_MK_LEN]
- u8 **k_aut** [EAP_SIM_K_AUT_LEN]
- u8 **k_encr** [EAP_SIM_K_ENCR_LEN]
- u8 **msk** [EAP_SIM_KEYING_DATA_LEN]
- u8 **emsk** [EAP_EMSK_LEN]
- u8 **rand** [3][GSM_RAND_LEN]
- int **num_id_req**
- int **num_notification**
- u8 ∗ **pseudonym**
- size_t **pseudonym_len**
- u8 ∗ **reauth_id**
- size_t **reauth_id_len**
- int **reauth**
- unsigned int **counter**
- unsigned int **counter_too_small**
- u8 ∗ **last_eap_identity**
- size_t **last_eap_identity_len**
- enum eap_sim_data:: { ... } **state**
- int **result_ind**
- int **use_result_ind**
- int **num_chal**
- enum eap_sim_data:: { ... } **state**
- char ∗ **next_pseudonym**
- char ∗ **next_reauth_id**
- u16 **counter**
- struct eap_sim_reauth ∗ **reauth**
- u16 **notification**

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

- src/eap_peer/eap_sim.c
- src/eap_server/eap_sim.c

14.83 eap_sim_db_data Struct Reference

Data Fields

- int **sock**
- char * **fname**
- char * **local_sock**
- void(* **get_complete_cb**)(void *ctx, void *session_ctx)
- void * **ctx**
- struct eap_sim_pseudonym * **pseudonyms**
- struct eap_sim_reauth * **reauths**
- struct eap_sim_db_pending * **pending**

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

- src/eap_server/eap_sim_db.c

14.84 eap_sim_db_pending Struct Reference

Public Types

- enum { **PENDING**, **SUCCESS**, **FAILURE** }

Data Fields

- struct eap_sim_db_pending * **next**
- u8 **imsi** [20]
- size_t **imsi_len**
- enum eap_sim_db_pending:: { ... } **state**
- void * **cb_session_ctx**
- struct os_time **timestamp**
- int **aka**
- union {
  struct {
  u8 **kc** [EAP_SIM_MAX_CHAL][EAP_SIM_KC_LEN]
  u8 **sres** [EAP_SIM_MAX_CHAL][EAP_SIM_SRES_LEN]
  u8 **rand** [EAP_SIM_MAX_CHAL][GSM_RAND_LEN]
  int **num_chal**
  } **sim**
  struct {
  u8 **rand** [EAP_AKA_RAND_LEN]
  u8 **autn** [EAP_AKA_AUTN_LEN]
  u8 **ik** [EAP_AKA_IK_LEN]
  u8 **ck** [EAP_AKA_CK_LEN]
  u8 **res** [EAP_AKA_RES_MAX_LEN]
  size_t **res_len**
  } **aka**
  } **u**

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

- src/eap_server/eap_sim_db.c

14.85 eap_sim_msg Struct Reference

Data Fields

- struct wpabuf * **buf**
- size_t **mac**
- size_t **iv**
- size_t **encr**
- int **type**

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

- src/eap_common/eap_sim_common.c

14.86 eap_sim_pseudonym Struct Reference

Data Fields

- struct eap_sim_pseudonym * **next**
- u8 * **identity**
- size_t **identity_len**
- char * **pseudonym**

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

- src/eap_server/eap_sim_db.c

14.87 eap_sm Struct Reference

EAP state machine data.

`#include <eap_i.h>`

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 }
- enum {
  EAP_DISABLED, **EAP_INITIALIZE**, **EAP_IDLE**, **EAP_RECEIVED**,
  EAP_INTEGRITY_CHECK, **EAP_METHOD_RESPONSE**, **EAP_METHOD_REQUEST**, **EAP_PROPOSE_METHOD**,
  EAP_SELECT_ACTION, **EAP_SEND_REQUEST**, **EAP_DISCARD**, **EAP_NAK**,
  EAP_RETRANSMIT, **EAP_SUCCESS**, **EAP_FAILURE**, **EAP_TIMEOUT_FAILURE**,
  EAP_PICK_UP_METHOD, **EAP_INITIALIZE_PASSTHROUGH**, **EAP_IDLE2**, **EAP_-RETRANSMIT2**,
  EAP_RECEIVED2, **EAP_DISCARD2**, **EAP_SEND_REQUEST2**, **EAP_AAA_REQUEST**,
  EAP_AAA_RESPONSE, **EAP_AAA_IDLE**, **EAP_TIMEOUT_FAILURE2**, **EAP_FAILURE2**,
  EAP_SUCCESS2 }
- enum { **METHOD_PROPOSED**, **METHOD_CONTINUE**, **METHOD_END** }
- enum { **DECISION_SUCCESS**, **DECISION_FAILURE**, **DECISION_CONTINUE**, **DECISION_PASSTHROUGH** }
- enum { **METHOD_PENDING_NONE**, **METHOD_PENDING_WAIT**, **METHOD_-PENDING_CONT** }
- enum { **NO_PROV**, **ANON_PROV**, **AUTH_PROV**, **BOTH_PROV** }

Data Fields

- enum eap_sm:: { ... } **EAP_state**
- EapType **selectedMethod**
- EapMethodState **methodState**
- int **lastId**
- struct wpabuf * **lastRespData**
- EapDecision **decision**
- Boolean **rxReq**
- Boolean **rxSuccess**
- Boolean **rxFailure**
- int **reqId**
- EapType **reqMethod**
- int **reqVendor**
- u32 **reqVendorMethod**
- Boolean **ignore**

- int **ClientTimeout**
- Boolean **allowNotifications**
- struct wpabuf * **eapRespData**
- Boolean **eapKeyAvailable**
- u8 * **eapKeyData**
- size_t **eapKeyDataLen**
- struct eap_method * **m**
- Boolean **changed**
- void * **eapol_ctx**
- struct 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**
- struct wps_context * **wps**
- int **prev_failure**
- enum eap_sm:: { ... } **EAP_state**
- int **MaxRetrans**
- struct eap_eapol_interface **eap_if**
- EapType **currentMethod**
- int **currentId**
- enum eap_sm:: { ... } **methodState**
- int **retransCount**
- struct wpabuf * **lastReqData**
- int **methodTimeout**
- int **respId**
- EapType **respMethod**
- int **respVendor**
- u32 **respVendorMethod**
- enum eap_sm:: { ... } **decision**
- u8 * **identity**
- size_t **identity_len**
- int **require_identity_match**
- struct eap_user * **user**
- int **user_eap_method_index**
- void * **eap_sim_db_priv**
- Boolean **backend_auth**
- Boolean **update_user**

- int **eap_server**
- enum eap_sm:: { ... } **method_pending**
- u8 * **pac_opaque_encr_key**
- u8 * **eap_fast_a_id**
- size_t **eap_fast_a_id_len**
- char * **eap_fast_a_id_info**
- enum eap_sm:: { ... } **eap_fast_prov**
- int **pac_key_lifetime**
- int **pac_key_refresh_time**
- int **eap_sim_aka_result_ind**
- int **tnc**
- struct wpabuf * **assoc_wps_ie**
- Boolean **start_reauth**
- u8 **peer_addr** [ETH_ALEN]

14.87.1 Detailed Description

EAP state machine data. EAP server state machine data.

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

- src/eap_peer/eap_i.h
- src/eap_server/eap_i.h

14.88 eap_ssl_data Struct Reference

TLS data for EAP methods.

`#include <eap_tls_common.h>`

Public Types

- enum { **MSG**, **FRAG_ACK**, **WAIT_FRAG_ACK** }

Data Fields

- struct tls_connection * conn

  TLS connection context data from tls_connection_init().

- u8 * tls_out

  TLS message to be sent out in fragments.

- size_t tls_out_len

  Total length of the outgoing TLS message.

- size_t tls_out_pos

  The current position in the outgoing TLS message.

- size_t tls_out_limit

  Maximum fragment size for outgoing TLS messages.

- u8 * tls_in

  Received TLS message buffer for re-assembly.

- size_t tls_in_len

  Number of bytes of the received TLS message in tls_in.

- size_t tls_in_left

  Number of remaining bytes in the incoming TLS message.

- size_t tls_in_total

  Total number of bytes in the incoming TLS message.

- int phase2

  Whether this TLS connection is used in EAP phase 2 (tunnel).

- int include_tls_length

  Whether the TLS length field is included even.

- int tls_ia

  Whether TLS/IA is enabled for this TLS connection.

- struct eap_sm * eap

Pointer to EAP state machine allocated with eap_peer_sm_init().

- enum eap_ssl_data:: { ... } **state**
- struct wpabuf * **in_buf**
- struct wpabuf * **out_buf**
- size_t **out_used**
- struct wpabuf **tmpbuf**

14.88.1 Detailed Description

TLS data for EAP methods.

14.88.2 Field Documentation

14.88.2.1 int eap_ssl_data::include_tls_length

Whether the TLS length field is included even. if the TLS data is not fragmented

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

- src/eap_peer/eap_tls_common.h
- src/eap_server/eap_tls_common.h

14.89 eap_tls_data Struct Reference

Public Types

- enum { **START**, **CONTINUE**, **SUCCESS**, **FAILURE** }

Data Fields

- struct eap_ssl_data **ssl**
- u8 * **key_data**
- enum eap_tls_data:: { ... } **state**
- int **established**

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

- src/eap_peer/eap_tls.c
- src/eap_server/eap_tls.c

14.90 eap_tlv_crypto_binding_tlv Struct Reference

Data Fields

- be16 **tlv_type**
- be16 **length**
- u8 **reserved**
- u8 **version**
- u8 **received_version**
- u8 **subtype**
- u8 **nonce** [32]
- u8 **compound_mac** [20]

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

- src/eap_common/eap_tlv_common.h

14.91 eap_tlv_hdr Struct Reference

Data Fields

- be16 **tlv_type**
- be16 **length**

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

- src/eap_common/eap_tlv_common.h

14.92 eap_tlv_intermediate_result_tlv Struct Reference

Data Fields

- be16 **tlv_type**
- be16 **length**
- be16 **status**

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

- src/eap_common/eap_tlv_common.h

14.93 eap_tlv_nak_tlv Struct Reference

Data Fields

- be16 **tlv_type**
- be16 **length**
- be32 **vendor_id**
- be16 **nak_type**

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

- src/eap_common/eap_tlv_common.h

14.94 eap_tlv_pac_ack_tlv Struct Reference

Data Fields

- be16 **tlv_type**
- be16 **length**
- be16 **pac_type**
- be16 **pac_len**
- be16 **result**

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

- src/eap_common/eap_tlv_common.h

14.95 eap_tlv_pac_type_tlv Struct Reference

Data Fields

- be16 **tlv_type**
- be16 **length**
- be16 **pac_type**

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

- src/eap_common/eap_tlv_common.h

14.96 eap_tlv_request_action_tlv Struct Reference

Data Fields

- be16 **tlv_type**
- be16 **length**
- be16 **action**

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

- src/eap_common/eap_tlv_common.h

14.97 eap_tlv_result_tlv Struct Reference

Data Fields

- be16 **tlv_type**
- be16 **length**
- be16 **status**

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

- src/eap_common/eap_tlv_common.h

14.98 eap_tnc_data Struct Reference

Public Types

- enum {
  WAIT_START, **PROC_MSG**, **WAIT_FRAG_ACK**, **DONE**,
  FAIL }
- enum {
  START, **CONTINUE**, **RECOMMENDATION**, **FRAG_ACK**,
  WAIT_FRAG_ACK, **DONE**, **FAIL** }
- enum { **ALLOW**, **ISOLATE**, **NO_ACCESS**, **NO_RECOMMENDATION** }

Data Fields

- enum eap_tnc_data:: { ... } **state**
- struct tncc_data * **tncc**
- struct wpabuf * **in_buf**
- struct wpabuf * **out_buf**
- size_t **out_used**
- size_t **fragment_size**
- enum eap_tnc_data:: { ... } **state**
- enum eap_tnc_data:: { ... } **recommendation**
- struct tncs_data * **tncs**

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

- src/eap_peer/eap_tnc.c
- src/eap_server/eap_tnc.c

14.99 eap_ttls_avp Struct Reference

Data Fields

- u8 * **eap**
- size_t **eap_len**
- u8 * **user_name**
- size_t **user_name_len**
- u8 * **user_password**
- size_t **user_password_len**
- u8 * **chap_challenge**
- size_t **chap_challenge_len**
- u8 * **chap_password**
- size_t **chap_password_len**
- u8 * **mschap_challenge**
- size_t **mschap_challenge_len**
- u8 * **mschap_response**
- size_t **mschap_response_len**
- u8 * **mschap2_response**
- size_t **mschap2_response_len**

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

- src/eap_server/eap_ttls.c

14.100 eap_ttls_data Struct Reference

Public Types

- enum **phase2_types** {

  EAP_TTLS_PHASE2_EAP, **EAP_TTLS_PHASE2_MSCHAPV2**, **EAP_TTLS_PHASE2_-MSCHAP**, **EAP_TTLS_PHASE2_PAP**,

  EAP_TTLS_PHASE2_CHAP }
- enum {

  START, **PHASE1**, **PHASE2_START**, **PHASE2_METHOD**,

  PHASE2_MSCHAPV2_RESP, **PHASE_FINISHED**, **SUCCESS**, **FAILURE** }

Data Fields

- struct eap_ssl_data **ssl**
- int **ssl_initialized**
- int **ttls_version**
- int **force_ttls_version**
- struct eap_method * **phase2_method**
- void * **phase2_priv**
- int **phase2_success**
- int **phase2_start**
- enum eap_ttls_data::phase2_types **phase2_type**
- struct eap_method_type **phase2_eap_type**
- struct eap_method_type * **phase2_eap_types**
- size_t **num_phase2_eap_types**
- u8 **auth_response** [MSCHAPV2_AUTH_RESPONSE_LEN]
- int **auth_response_valid**
- u8 **master_key** [MSCHAPV2_MASTER_KEY_LEN]
- u8 **ident**
- int **resuming**
- int **reauth**
- u8 * **key_data**
- struct wpabuf * **pending_phase2_req**
- enum eap_ttls_data:: { ... } **state**
- int **force_version**
- int **mschapv2_resp_ok**
- u8 **mschapv2_auth_response** [20]
- u8 **mschapv2_ident**
- int **tls_ia_configured**
- struct wpabuf * **pending_phase2_eap_resp**
- int **tnc_started**

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

- src/eap_peer/eap_ttls.c
- src/eap_server/eap_ttls.c

14.101 eap_user Struct Reference

Data Fields

- struct {
  int **vendor**
  u32 **method**
  } **methods** [EAP_MAX_METHODS]

- u8 * **password**
- size_t **password_len**
- int **password_hash**
- int **phase2**
- int **force_version**
- int **ttls_auth**

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

- src/eap_server/eap.h

14.102 eap_vendor_test_data Struct Reference

Public Types

- enum { **INIT**, **CONFIRM**, **SUCCESS** }
- enum { **INIT**, **CONFIRM**, **SUCCESS**, **FAILURE** }

Data Fields

- enum eap_vendor_test_data:: { ... } **state**
- int **first_try**
- enum eap_vendor_test_data:: { ... } **state**

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

- src/eap_peer/eap_vendor_test.c
- src/eap_server/eap_vendor_test.c

14.103 eap_wsc_data Struct Reference

Public Types

- enum {
  WAIT_START, **MESG**, **FRAG_ACK**, **WAIT_FRAG_ACK**,
  DONE, **FAIL** }
- enum {
  START, **MSG**, **FRAG_ACK**, **WAIT_FRAG_ACK**,
  DONE, **FAIL** }

Data Fields

- enum eap_wsc_data:: { ... } **state**
- int **registrar**
- struct wpabuf * **in_buf**
- struct wpabuf * **out_buf**
- enum wsc_op_code in_op_code **out_op_code**
- size_t **out_used**
- size_t **fragment_size**
- struct wps_data * **wps**
- struct wps_context * **wps_ctx**
- enum eap_wsc_data:: { ... } **state**
- int **ext_reg_timeout**

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

- src/eap_peer/eap_wsc.c
- src/eap_server/eap_wsc.c

14.104 eapol_auth_cb Struct Reference

Data Fields

- void(∗ **eapol_send**)(void ∗ctx, void ∗sta_ctx, u8 type, const u8 ∗data, size_t datalen)
- void(∗ **aaa_send**)(void ∗ctx, void ∗sta_ctx, const u8 ∗data, size_t datalen)
- void(∗ **finished**)(void ∗ctx, void ∗sta_ctx, int success, int preauth)
- int(∗ **get_eap_user**)(void ∗ctx, const u8 ∗identity, size_t identity_len, int phase2, struct eap_user ∗user)
- int(∗ **sta_entry_alive**)(void ∗ctx, const u8 ∗addr)
- void(∗ **logger**)(void ∗ctx, const u8 ∗addr, eapol_logger_level level, const char ∗txt)
- void(∗ **set_port_authorized**)(void ∗ctx, void ∗sta_ctx, int authorized)
- void(∗ **abort_auth**)(void ∗ctx, void ∗sta_ctx)
- void(∗ **tx_key**)(void ∗ctx, void ∗sta_ctx)

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

- hostapd/eapol_sm.h

14.105 eapol_auth_config Struct Reference

Data Fields

- int **eap_reauth_period**
- int **wpa**
- int **individual_wep_key_len**
- int **eap_server**
- void * **ssl_ctx**
- void * **eap_sim_db_priv**
- char * **eap_req_id_text**
- size_t **eap_req_id_text_len**
- u8 * **pac_opaque_encr_key**
- u8 * **eap_fast_a_id**
- size_t **eap_fast_a_id_len**
- char * **eap_fast_a_id_info**
- int **eap_fast_prov**
- int **pac_key_lifetime**
- int **pac_key_refresh_time**
- int **eap_sim_aka_result_ind**
- int **tnc**
- struct wps_context * **wps**
- struct hostapd_data * **hapd**

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

- hostapd/eapol_sm.h

14.106 eapol_authenticator Struct Reference

Global EAPOL authenticator data.

`#include <eapol_sm.h>`

Data Fields

- struct eapol_auth_config **conf**
- struct eapol_auth_cb **cb**
- u8 * **default_wep_key**
- u8 **default_wep_key_idx**

14.106.1 Detailed Description

Global EAPOL authenticator data.

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

- hostapd/eapol_sm.h

14.107 eapol_callbacks Struct Reference

Callback functions from EAP to lower layer.

`#include <eap.h>`

Data Fields

- struct eap_peer_config *(* 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.

- struct wpabuf *(* get_eapReqData)(void *ctx)

  Get EAP-Request data.

- void(* set_config_blob)(void *ctx, struct wpa_config_blob *blob)

  Set named configuration blob.

- struct wpa_config_blob *(* get_config_blob)(void *ctx, const char *name)

  Get a named configuration blob.

- void(* notify_pending)(void *ctx)

  Notify that a pending request can be retried.

- void(* eap_param_needed)(void *ctx, const char *field, const char *txt)

  Notify that EAP parameter is needed.

- int(* **get_eap_user**)(void *ctx, const u8 *identity, size_t identity_len, int phase2, struct eap_user *user)
- const char *(* **get_eap_req_id_text**)(void *ctx, size_t *len)

14.107.1 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_peer_sm_init() call will be used as the ctx parameter for these callback functions.

14.107.2 Field Documentation

14.107.2.1 void(∗ eapol_callbacks::eap_param_needed)(void ∗ctx, const char ∗field, const char ∗txt)

Notify that EAP parameter is needed.

Parameters:

ctx eapol_ctx from eap_peer_sm_init() call

field Field name (e.g., "IDENTITY")

txt User readable text describing the required parameter

14.107.2.2 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

14.107.2.3 struct eap_peer_config∗(∗ eapol_callbacks::get_config)(void ∗ctx) [read]

Get pointer to the current network configuration.

Parameters:

ctx eapol_ctx from eap_peer_sm_init() call

14.107.2.4 struct wpa_config_blob∗(∗ eapol_callbacks::get_config_blob)(void ∗ctx, const char ∗name) [read]

Get a named configuration blob.

Parameters:

ctx eapol_ctx from eap_peer_sm_init() call

name Name of the blob

Returns:

Pointer to blob data or NULL if not found

14.107.2.5 struct wpabuf*(* eapol_callbacks::get_eapReqData)(void *ctx) [read]

Get EAP-Request data.

Parameters:

ctx eapol_ctx from eap_peer_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.

14.107.2.6 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_peer_sm_init() call

variable EAPOL integer variable to get

Returns:

Value of the EAPOL variable

14.107.2.7 void(* eapol_callbacks::notify_pending)(void *ctx)

Notify that a pending request can be retried.

Parameters:

ctx eapol_ctx from eap_peer_sm_init() call

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

14.107.2.8 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_peer_sm_init() call

variable EAPOL boolean variable to set

value Value for the EAPOL variable

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

Set named configuration blob.

Parameters:

ctx eapol_ctx from eap_peer_sm_init() call

blob New value for the blob

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

14.107.2.10 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_peer_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 files:

- src/eap_peer/eap.h
- src/eap_server/eap.h

14.108 eapol_config Struct Reference

Per network configuration for EAPOL state machines.

`#include <eapol_supp_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.

14.108.1 Detailed Description

Per network configuration for EAPOL state machines.

14.108.2 Field Documentation

14.108.2.1 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.

14.108.2.2 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.

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

- src/eapol_supp/eapol_supp_sm.h

14.109 eapol_ctx Struct Reference

Global (for all networks) EAPOL state machine context.

`#include <eapol_supp_sm.h>`

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.
- 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.
- struct wps_context * wps
  WPS context data.

- void(∗ eap_param_needed)(void ∗ctx, const char ∗field, const char ∗txt)

  Notify that EAP parameter is needed.

- void(∗ port_cb)(void ∗ctx, int authorized)

  Set port authorized/unauthorized callback (optional).

14.109.1 Detailed Description

Global (for all networks) EAPOL state machine context.

14.109.2 Field Documentation

14.109.2.1 void(∗ eapol_ctx::aborted_cached)(void ∗ctx)

Notify that cached PMK attempt was aborted.

Parameters:

ctx Callback context (ctx)

14.109.2.2 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.

14.109.2.3 void(∗ eapol_ctx::eap_param_needed)(void ∗ctx, const char ∗field, const char ∗txt)

Notify that EAP parameter is needed.

Parameters:

ctx Callback context (ctx)

field Field name (e.g., "IDENTITY")

txt User readable text describing the required parameter

14.109.2.4 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.

14.109.2.5 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

14.109.2.6 struct wpa_config_blob∗(∗ eapol_ctx::get_config_blob)(void ∗ctx, const char ∗name) [read]

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

14.109.2.7 void(∗ eapol_ctx::port_cb)(void ∗ctx, int authorized)

Set port authorized/unauthorized callback (optional).

Parameters:

ctx Callback context (ctx)

authorized Whether the supplicant port is now in authorized state

14.109.2.8 int eapol_ctx::preauth

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

14.109.2.9 void* eapol_ctx::scard_ctx

Callback context for PC/SC scard_*() function calls. This context can be updated with eapol_sm_register_scard_ctx().

14.109.2.10 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.

14.109.2.11 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

14.109.2.12 struct wps_context* eapol_ctx::wps [read]

WPS context data. This is only used by EAP-WSC and can be left NULL if not available.

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

- src/eapol_supp/eapol_supp_sm.h

14.110 eapol_sm Struct Reference

Internal data for EAPOL state machines.

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**
- int **timer_tick_enabled**
- Boolean **eapFail**
- Boolean **eapolEap**
- Boolean **eapSuccess**
- Boolean **initialize**
- Boolean **keyDone**
- Boolean **keyRun**
- PortControl **portControl**
- Boolean **portEnabled**
- PortStatus **suppPortStatus**
- Boolean **portValid**
- Boolean **suppAbort**
- Boolean **suppFail**
- Boolean **suppStart**
- Boolean **suppSuccess**
- Boolean **suppTimeout**
- enum eapol_sm:: { ... } **SUPP_PAE_state**
- Boolean **userLogoff**
- Boolean **logoffSent**
- unsigned int **startCount**

- Boolean **eapRestart**
- PortControl **sPortMode**
- unsigned int **heldPeriod**
- unsigned int **startPeriod**
- unsigned int **maxStart**
- enum eapol_sm:: { ... } **KEY_RX_state**
- Boolean **rxKey**
- enum eapol_sm:: { ... } **SUPP_BE_state**
- Boolean **eapNoResp**
- Boolean **eapReq**
- Boolean **eapResp**
- unsigned int **authPeriod**
- unsigned int **dot1xSuppEapolFramesRx**
- unsigned int **dot1xSuppEapolFramesTx**
- unsigned int **dot1xSuppEapolStartFramesTx**
- unsigned int **dot1xSuppEapolLogoffFramesTx**
- unsigned int **dot1xSuppEapolRespFramesTx**
- unsigned int **dot1xSuppEapolReqIdFramesRx**
- unsigned int **dot1xSuppEapolReqFramesRx**
- unsigned int **dot1xSuppInvalidEapolFramesRx**
- unsigned int **dot1xSuppEapLengthErrorFramesRx**
- unsigned int **dot1xSuppLastEapolFrameVersion**
- unsigned char **dot1xSuppLastEapolFrameSource** [6]
- Boolean **changed**
- struct eap_sm * **eap**
- struct eap_peer_config * **config**
- Boolean **initial_req**
- u8 * **last_rx_key**
- size_t **last_rx_key_len**
- struct wpabuf * **eapReqData**
- Boolean **altAccept**
- Boolean **altReject**
- Boolean **replay_counter_valid**
- u8 **last_replay_counter** [16]
- struct eapol_config **conf**
- struct eapol_ctx * **ctx**
- enum eapol_sm:: { ... } **cb_status**
- Boolean **cached_pmk**
- Boolean **unicast_key_received**
- Boolean **broadcast_key_received**

14.110.1 Detailed Description

Internal data for EAPOL state machines.

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

- src/eapol_supp/eapol_supp_sm.c

14.111 eapol_state_machine Struct Reference

Per-Supplicant Authenticator state machines.

`#include <eapol_sm.h>`

Public Types

- enum {

  AUTH_PAE_INITIALIZE, **AUTH_PAE_DISCONNECTED**, **AUTH_PAE_CONNECTING**, **AUTH_PAE_AUTHENTICATING**,

  AUTH_PAE_AUTHENTICATED, **AUTH_PAE_ABORTING**, **AUTH_PAE_HELD**, **AUTH_PAE_FORCE_AUTH**,

  AUTH_PAE_FORCE_UNAUTH, **AUTH_PAE_RESTART** }
- enum {

  BE_AUTH_REQUEST, **BE_AUTH_RESPONSE**, **BE_AUTH_SUCCESS**, **BE_AUTH_FAIL**,

  BE_AUTH_TIMEOUT, **BE_AUTH_IDLE**, **BE_AUTH_INITIALIZE**, **BE_AUTH_IGNORE** }
- enum { **REAUTH_TIMER_INITIALIZE**, **REAUTH_TIMER_REAUTHENTICATE** }
- enum { **AUTH_KEY_TX_NO_KEY_TRANSMIT**, **AUTH_KEY_TX_KEY_TRANSMIT** }
- enum { **KEY_RX_NO_KEY_RECEIVE**, **KEY_RX_KEY_RECEIVE** }
- enum { **CTRL_DIR_FORCE_BOTH**, **CTRL_DIR_IN_OR_BOTH** }

Data Fields

- int **aWhile**
- int **quietWhile**
- int **reAuthWhen**
- Boolean **authAbort**
- Boolean **authFail**
- PortState **authPortStatus**
- Boolean **authStart**
- Boolean **authTimeout**
- Boolean **authSuccess**
- Boolean **eapolEap**
- Boolean **initialize**
- Boolean **keyDone**
- Boolean **keyRun**
- Boolean **keyTxEnabled**
- PortTypes **portControl**
- Boolean **portValid**
- Boolean **reAuthenticate**
- enum eapol_state_machine:: { ... } **auth_pae_state**
- Boolean **eapolLogoff**
- Boolean **eapolStart**
- PortTypes **portMode**
- unsigned int **reAuthCount**
- unsigned int **quietPeriod**
- unsigned int **reAuthMax**
- Counter **authEntersConnecting**

- Counter **authEapLogoffsWhileConnecting**
- Counter **authEntersAuthenticating**
- Counter **authAuthSuccessesWhileAuthenticating**
- Counter **authAuthTimeoutsWhileAuthenticating**
- Counter **authAuthFailWhileAuthenticating**
- Counter **authAuthEapStartsWhileAuthenticating**
- Counter **authAuthEapLogoffWhileAuthenticating**
- Counter **authAuthReauthsWhileAuthenticated**
- Counter **authAuthEapStartsWhileAuthenticated**
- Counter **authAuthEapLogoffWhileAuthenticated**
- enum eapol_state_machine:: { ... } **be_auth_state**
- unsigned int **serverTimeout**
- Counter **backendResponses**
- Counter **backendAccessChallenges**
- Counter **backendOtherRequestsToSupplicant**
- Counter **backendAuthSuccesses**
- Counter **backendAuthFails**
- enum eapol_state_machine:: { ... } **reauth_timer_state**
- unsigned int **reAuthPeriod**
- Boolean **reAuthEnabled**
- enum eapol_state_machine:: { ... } **auth_key_tx_state**
- enum eapol_state_machine:: { ... } **key_rx_state**
- Boolean **rxKey**
- enum eapol_state_machine:: { ... } **ctrl_dir_state**
- ControlledDirection **adminControlledDirections**
- ControlledDirection **operControlledDirections**
- Boolean **operEdge**
- Counter **dot1xAuthEapolFramesRx**
- Counter **dot1xAuthEapolFramesTx**
- Counter **dot1xAuthEapolStartFramesRx**
- Counter **dot1xAuthEapolLogoffFramesRx**
- Counter **dot1xAuthEapolRespIdFramesRx**
- Counter **dot1xAuthEapolRespFramesRx**
- Counter **dot1xAuthEapolReqIdFramesTx**
- Counter **dot1xAuthEapolReqFramesTx**
- Counter **dot1xAuthInvalidEapolFramesRx**
- Counter **dot1xAuthEapLengthErrorFramesRx**
- Counter **dot1xAuthLastEapolFrameVersion**
- u8 **addr** [ETH_ALEN]
- int **flags**
- struct eap_eapol_interface * **eap_if**
- int **radius_identifier**
- struct radius_msg * **last_recv_radius**
- u8 **last_eap_id**
- u8 * **identity**
- size_t **identity_len**
- u8 **eap_type_authsrv**
- u8 **eap_type_supp**
- struct radius_class_data **radius_class**
- u8 * **eapol_key_sign**

- size_t **eapol_key_sign_len**
- u8 * **eapol_key_crypt**
- size_t **eapol_key_crypt_len**
- struct eap_sm * **eap**
- Boolean **initializing**
- Boolean **changed**
- struct eapol_authenticator * **eapol**
- struct hostapd_data * **hapd**
- struct sta_info * **sta**

14.111.1 Detailed Description

Per-Supplicant Authenticator state machines.

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

- hostapd/eapol_sm.h

14.112 eapol_test_data Struct Reference

Data Fields

- struct wpa_supplicant * **wpa_s**
- int **eapol_test_num_reauths**
- int **no_mppe_keys**
- int **num_mppe_ok**
- int **num_mppe_mismatch**
- u8 **radius_identifier**
- struct radius_msg * **last_recv_radius**
- struct in_addr **own_ip_addr**
- struct radius_client_data * **radius**
- struct hostapd_radius_servers * **radius_conf**
- u8 * **last_eap_radius**
- size_t **last_eap_radius_len**
- u8 **authenticator_pmk** [PMK_LEN]
- size_t **authenticator_pmk_len**
- int **radius_access_accept_received**
- int **radius_access_reject_received**
- int **auth_timed_out**
- u8 * **eap_identity**
- size_t **eap_identity_len**
- char * **connect_info**
- u8 **own_addr** [ETH_ALEN]
- struct extra_radius_attr * **extra_attrs**

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

- wpa_supplicant/eapol_test.c

14.113 eloop_data Struct Reference

Data Fields

- void * **user_data**
- int **max_sock**
- struct eloop_sock_table **readers**
- struct eloop_sock_table **writers**
- struct eloop_sock_table **exceptions**
- struct eloop_timeout * **timeout**
- int **signal_count**
- struct eloop_signal * **signals**
- int **signaled**
- int **pending_terminate**
- int **terminate**
- int **reader_table_changed**
- int **reader_count**
- struct eloop_sock * **readers**
- size_t **reader_count**
- size_t **event_count**
- struct eloop_event * **events**
- struct eloop_signal **term_signal**
- HANDLE **term_event**
- HANDLE * **handles**
- size_t **num_handles**

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

- src/utils/eloop.c
- src/utils/eloop_none.c
- src/utils/eloop_win.c

14.114 eloop_event Struct Reference

Data Fields

- void * **eloop_data**
- void * **user_data**
- eloop_event_handler **handler**
- HANDLE **event**

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

- src/utils/eloop_win.c

14.115 eloop_signal Struct Reference

Data Fields

- int **sig**
- void * **user_data**
- eloop_signal_handler **handler**
- int **signaled**
- void(* **handler**)(int sig, void *eloop_ctx, void *signal_ctx)

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

- src/utils/eloop.c
- src/utils/eloop_none.c
- src/utils/eloop_win.c

14.116 eloop_sock Struct Reference

Data Fields

- int **sock**
- void * **eloop_data**
- void * **user_data**
- eloop_sock_handler **handler**
- void(* **handler**)(int sock, void *eloop_ctx, void *sock_ctx)
- WSAEVENT **event**

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

- src/utils/eloop.c
- src/utils/eloop_none.c
- src/utils/eloop_win.c

14.117 eloop_sock_table Struct Reference

Data Fields

- int **count**
- struct eloop_sock * **table**
- int **changed**

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

- src/utils/eloop.c

14.118 eloop_timeout Struct Reference

Data Fields

- struct os_time **time**
- void * **eloop_data**
- void * **user_data**
- eloop_timeout_handler **handler**
- struct eloop_timeout * **next**
- void(* **handler**)(void *eloop_ctx, void *sock_ctx)

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

- src/utils/eloop.c
- src/utils/eloop_none.c
- src/utils/eloop_win.c

14.119 extra_radius_attr Struct Reference

Data Fields

- u8 **type**
- char **syntax**
- char * **data**
- struct extra_radius_attr * **next**

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

- wpa_supplicant/eapol_test.c

14.120 family_data Struct Reference

Data Fields

- const char * **group**
- int **id**

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

- src/drivers/driver_nl80211.c

14.121 wpa_event_data::ft_ies Struct Reference

FT information elements (EVENT_FT_RESPONSE).

`#include <driver.h>`

Data Fields

- const u8 * **ies**
- size_t **ies_len**
- int **ft_action**
- u8 **target_ap** [ETH_ALEN]
- const u8 * ric_ies
- size_t ric_ies_len

14.121.1 Detailed Description

FT information elements (EVENT_FT_RESPONSE). During FT (IEEE 802.11r) authentication sequence, the driver is expected to use this event to report received FT IEs (MDIE, FTIE, RSN IE, TIE, possible resource request) to the supplicant. The FT IEs for the next message will be delivered through the struct wpa_driver_ops::update_ft_ies() callback.

14.121.2 Field Documentation

14.121.2.1 const u8* wpa_event_data::ft_ies::ric_ies

Optional IE(s), e.g., WMM TSPEC(s), for RIC-Request

14.121.2.2 size_t wpa_event_data::ft_ies::ric_ies_len

Length of ric_ies buffer in octets

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

- src/drivers/driver.h

14.122 ft_r0kh_r1kh_pull_frame Struct Reference

Data Fields

- u8 **frame_type**
- u8 **packet_type**
- le16 **data_length**
- u8 **ap_address** [ETH_ALEN]
- u8 **nonce** [16]
- u8 **pmk_r0_name** [WPA_PMK_NAME_LEN]
- u8 **r1kh_id** [FT_R1KH_ID_LEN]
- u8 **s1kh_id** [ETH_ALEN]
- u8 **pad** [4]
- u8 **key_wrap_extra** [8]

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

- hostapd/wpa.h

14.123 ft_r0kh_r1kh_push_frame Struct Reference

Data Fields

- u8 **frame_type**
- u8 **packet_type**
- le16 **data_length**
- u8 **ap_address** [ETH_ALEN]
- u8 **timestamp** [4]
- u8 **r1kh_id** [FT_R1KH_ID_LEN]
- u8 **s1kh_id** [ETH_ALEN]
- u8 **pmk_r0_name** [WPA_PMK_NAME_LEN]
- u8 **pmk_r1** [PMK_LEN]
- u8 **pmk_r1_name** [WPA_PMK_NAME_LEN]
- u8 **key_wrap_extra** [8]

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

- hostapd/wpa.h

14.124 ft_r0kh_r1kh_resp_frame Struct Reference

Data Fields

- u8 **frame_type**
- u8 **packet_type**
- le16 **data_length**
- u8 **ap_address** [ETH_ALEN]
- u8 **nonce** [16]
- u8 **r1kh_id** [FT_R1KH_ID_LEN]
- u8 **s1kh_id** [ETH_ALEN]
- u8 **pmk_r1** [PMK_LEN]
- u8 **pmk_r1_name** [WPA_PMK_NAME_LEN]
- u8 **pad** [4]
- u8 **key_wrap_extra** [8]

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

- hostapd/wpa.h

14.125 ft_remote_r0kh Struct Reference

Data Fields

- struct ft_remote_r0kh * **next**
- u8 **addr** [ETH_ALEN]
- u8 **id** [FT_R0KH_ID_MAX_LEN]
- size_t **id_len**
- u8 **key** [16]

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

- hostapd/wpa.h

14.126 ft_remote_r1kh Struct Reference

Data Fields

- struct ft_remote_r1kh * **next**
- u8 **addr** [ETH_ALEN]
- u8 **id** [FT_R1KH_ID_LEN]
- u8 **key** [16]

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

- hostapd/wpa.h

14.127 ft_rrb_frame Struct Reference

Data Fields

- u8 **frame_type**
- u8 **packet_type**
- le16 **action_length**
- u8 **ap_address** [ETH_ALEN]

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

- hostapd/wpa.h

14.128 global_parse_data Struct Reference

Data Fields

- char * **name**
- int(* **parser**)(const struct global_parse_data *data, struct wpa_config *config, int line, const char *value)
- void * **param1**
- void * **param2**
- void * **param3**

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

- wpa_supplicant/config_file.c

14.129 gnutls_session_int Struct Reference

Data Fields

- security_parameters_st **security_parameters**

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

- src/crypto/tls_gnutls.c

14.130 gsm_triplet Struct Reference

Data Fields

- struct gsm_triplet * **next**
- char **imsi** [20]
- u8 **kc** [8]
- u8 **sres** [4]
- u8 **_rand** [16]

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

- src/hlr_auc_gw/hlr_auc_gw.c

14.131 hapd_interfaces Struct Reference

Data Fields

- size_t **count**
- struct hostapd_iface ** **iface**

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

- hostapd/main.c

14.132 hostap_sta_driver_data Struct Reference

Data Fields

- unsigned long **rx_packets**
- unsigned long **tx_packets**
- unsigned long **rx_bytes**
- unsigned long **tx_bytes**
- unsigned long **current_tx_rate**
- unsigned long **inactive_msec**
- unsigned long **flags**
- unsigned long **num_ps_buf_frames**
- unsigned long **tx_retry_failed**
- unsigned long **tx_retry_count**
- int **last_rssi**
- int **last_ack_rssi**

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

- src/drivers/driver.h

14.133 hostapd_acl_query_data Struct Reference

Data Fields

- time_t **timestamp**
- u8 **radius_id**
- macaddr **addr**
- u8 * **auth_msg**
- size_t **auth_msg_len**
- struct hostapd_acl_query_data * **next**

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

- hostapd/ieee802_11_auth.c

14.134 hostapd_bss_config Struct Reference

Per-BSS configuration.

`#include <config.h>`

Public Types

- enum { **ACCEPT_UNLESS_DENIED** = 0, **DENY_UNLESS_ACCEPTED** = 1, **USE_EXTERNAL_RADIUS_AUTH** = 2 }

Data Fields

- char **iface** [IFNAMSIZ+1]
- char **bridge** [IFNAMSIZ+1]
- enum hostapd_logger_level logger_syslog_level **logger_stdout_level**
- unsigned int **logger_syslog**
- unsigned int **logger_stdout**
- char * **dump_log_name**
- int **max_num_sta**
- int **dtim_period**
- int **ieee802_1x**
- int **eapol_version**
- int **eap_server**
- struct hostapd_eap_user * **eap_user**
- char * **eap_sim_db**
- struct hostapd_ip_addr **own_ip_addr**
- char * **nas_identifier**
- struct hostapd_radius_servers * **radius**
- int **acct_interim_interval**
- struct hostapd_ssid **ssid**
- char * **eap_req_id_text**
- size_t **eap_req_id_text_len**
- int **eapol_key_index_workaround**
- size_t **default_wep_key_len**
- int **individual_wep_key_len**
- int **wep_rekeying_period**
- int **broadcast_key_idx_min**
- int **broadcast_key_idx_max**
- int **eap_reauth_period**
- int **ieee802_11f**
- char **iapp_iface** [IFNAMSIZ+1]
- enum hostapd_bss_config:: { ... } **macaddr_acl**
- struct mac_acl_entry * **accept_mac**
- int **num_accept_mac**
- struct mac_acl_entry * **deny_mac**
- int **num_deny_mac**
- int **auth_algs**
- int **wpa**
- int **wpa_key_mgmt**

- int **wpa_pairwise**
- int **wpa_group**
- int **wpa_group_rekey**
- int **wpa_strict_rekey**
- int **wpa_gmk_rekey**
- int **wpa_ptk_rekey**
- int **rsn_pairwise**
- int **rsn_preauth**
- char * **rsn_preauth_interfaces**
- int **peerkey**
- char * **ctrl_interface**
- gid_t **ctrl_interface_gid**
- int **ctrl_interface_gid_set**
- char * **ca_cert**
- char * **server_cert**
- char * **private_key**
- char * **private_key_passwd**
- int **check_crl**
- char * **dh_file**
- u8 * **pac_opaque_encr_key**
- u8 * **eap_fast_a_id**
- size_t **eap_fast_a_id_len**
- char * **eap_fast_a_id_info**
- int **eap_fast_prov**
- int **pac_key_lifetime**
- int **pac_key_refresh_time**
- int **eap_sim_aka_result_ind**
- int **tnc**
- char * **radius_server_clients**
- int **radius_server_auth_port**
- int **radius_server_ipv6**
- char * **test_socket**
- int **use_pae_group_addr**
- int **ap_max_inactivity**
- int **ignore_broadcast_ssid**
- int **wmm_enabled**
- struct hostapd_vlan * **vlan**
- struct hostapd_vlan * **vlan_tail**
- macaddr **bssid**
- u16 **max_listen_interval**
- int **okc**
- int **wps_state**

14.134.1 Detailed Description

Per-BSS configuration.

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

- hostapd/config.h

14.135 hostapd_cached_radius_acl Struct Reference

Data Fields

- time_t **timestamp**
- macaddr **addr**
- int **accepted**
- struct hostapd_cached_radius_acl * **next**
- u32 **session_timeout**
- u32 **acct_interim_interval**
- int **vlan_id**

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

- hostapd/ieee802_11_auth.c

14.136 hostapd_channel_data Struct Reference

Data Fields

- short **chan**
- short **freq**
- int **flag**
- u8 **max_tx_power**

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

- src/drivers/driver.h

14.137 hostapd_cli_cmd Struct Reference

Data Fields

- const char * **cmd**
- int(* **handler**)(struct wpa_ctrl *ctrl, int argc, char *argv[ ])

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

- hostapd/hostapd_cli.c

14.138 hostapd_config Struct Reference

Per-radio interface configuration.

`#include <config.h>`

Public Types

- enum { **LONG_PREAMBLE** = 0, **SHORT_PREAMBLE** = 1 }
- enum { **CTS_PROTECTION_AUTOMATIC** = 0, **CTS_PROTECTION_FORCE_ENABLED** = 1, **CTS_PROTECTION_FORCE_DISABLED** = 2, **CTS_PROTECTION_AUTOMATIC_NO_OLBC** = 3 }
- enum { **INTERNAL_BRIDGE_DO_NOT_CONTROL** = -1, **INTERNAL_BRIDGE_DISABLED** = 0, **INTERNAL_BRIDGE_ENABLED** = 1 }

Data Fields

- struct hostapd_bss_config * **bss**
- struct hostapd_bss_config * **last_bss**
- size_t **num_bss**
- u16 **beacon_int**
- int **rts_threshold**
- int **fragm_threshold**
- u8 **send_probe_response**
- u8 **channel**
- hostapd_hw_mode **hw_mode**
- enum hostapd_config:: { ... } **preamble**
- enum hostapd_config:: { ... } **cts_protection_type**
- int * **supported_rates**
- int * **basic_rates**
- struct wpa_driver_ops * **driver**
- int **ap_table_max_size**
- int **ap_table_expiration_time**
- char **country** [3]
- int **ieee80211d**
- struct hostapd_tx_queue_params **tx_queue** [NUM_TX_QUEUES]
- struct hostapd_wmm_ac_params **wmm_ac_params** [4]
- enum hostapd_config:: { ... } **bridge_packets**
- int **ieee80211n**
- int **secondary_channel**

14.138.1 Detailed Description

Per-radio interface configuration.

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

- hostapd/config.h

14.139 hostapd_data Struct Reference

hostapd per-BSS data structure

`#include <hostapd.h>`

Data Fields

- struct hostapd_iface * **iface**
- struct hostapd_config * **iconf**
- struct hostapd_bss_config * **conf**
- int **interface_added**
- u8 **own_addr** [ETH_ALEN]
- int **num_sta**
- struct sta_info * **sta_list**
- struct sta_info * **sta_hash** [STA_HASH_SIZE]
- u32 **sta_aid** [AID_WORDS]
- struct wpa_driver_ops * **driver**
- void * **drv_priv**
- void * **msg_ctx**
- struct radius_client_data * **radius**
- int **radius_client_reconfigured**
- u32 **acct_session_id_hi**
- u32 **acct_session_id_lo**
- struct iapp_data * **iapp**
- struct hostapd_cached_radius_acl * **acl_cache**
- struct hostapd_acl_query_data * **acl_queries**
- struct wpa_authenticator * **wpa_auth**
- struct eapol_authenticator * **eapol_auth**
- struct rsn_preauth_interface * **preauth_iface**
- time_t **michael_mic_failure**
- int **michael_mic_failures**
- int **tkip_countermeasures**
- int **ctrl_sock**
- struct wpa_ctrl_dst * **ctrl_dst**
- void * **ssl_ctx**
- void * **eap_sim_db_priv**
- struct radius_server_data * **radius_srv**
- int **parameter_set_count**
- struct l2_packet_data * **l2**
- struct wps_context * **wps**
- struct hostapd_probereq_cb * **probereq_cb**
- size_t **num_probereq_cb**

14.139.1 Detailed Description

hostapd per-BSS data structure

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

- hostapd/hostapd.h

14.140 hostapd_eap_user Struct Reference

Data Fields

- struct hostapd_eap_user * **next**
- u8 * **identity**
- size_t **identity_len**
- struct {
  int **vendor**
  u32 **method**
  } **methods** [EAP_USER_MAX_METHODS]

- u8 * **password**
- size_t **password_len**
- int **phase2**
- int **force_version**
- unsigned int **wildcard_prefix**:1
- unsigned int **password_hash**:1
- int **ttls_auth**

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

- hostapd/config.h

14.141 hostapd_frame_info Struct Reference

Data Fields

- u32 **phytype**
- u32 **channel**
- u32 **datarate**
- u32 **ssi_signal**

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

- src/drivers/driver.h

14.142 hostapd_freq_params Struct Reference

Data Fields

- int **mode**
- int **freq**
- int **channel**
- int **ht_enabled**
- int **sec_channel_offset**

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

- src/drivers/driver.h

14.143 hostapd_hw_modes Struct Reference

Data Fields

- hostapd_hw_mode **mode**
- int **num_channels**
- struct hostapd_channel_data * **channels**
- int **num_rates**
- struct hostapd_rate_data * **rates**
- u16 **ht_capab**
- u8 **mcs_set** [16]

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

- src/drivers/driver.h

14.144 hostapd_iface Struct Reference

hostapd per-interface data structure

`#include <hostapd.h>`

Data Fields

- void * **owner**
- char * **config_fname**
- struct hostapd_config * **conf**
- size_t **num_bss**
- struct hostapd_data ** **bss**
- int **num_ap**
- struct ap_info * **ap_list**
- struct ap_info * **ap_hash** [STA_HASH_SIZE]
- struct ap_info * **ap_iter_list**
- struct hostapd_hw_modes * **hw_features**
- int **num_hw_features**
- struct hostapd_hw_modes * **current_mode**
- int **num_rates**
- struct hostapd_rate_data * **current_rates**
- u16 **hw_flags**
- int **num_sta_non_erp**
- int **num_sta_no_short_slot_time**
- int **num_sta_no_short_preamble**
- int **olbc**
- int **num_sta_ht_no_gf**
- int **num_sta_no_ht**
- int **num_sta_ht_20mhz**
- int **olbc_ht**
- u16 **ht_op_mode**
- void(* **scan_cb**)(struct hostapd_iface *iface)

14.144.1 Detailed Description

hostapd per-interface data structure

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

- hostapd/hostapd.h

14.145 hostapd_ip_addr Struct Reference

Data Fields

- union {
  struct in_addr **v4**
  } **u**

- int **af**

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

- src/utils/ip_addr.h

14.146 hostapd_probereq_cb Struct Reference

Data Fields

- void(∗ **cb**)(void ∗ctx, const u8 ∗sa, const u8 ∗ie, size_t ie_len)
- void ∗ **ctx**

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

- hostapd/hostapd.h

14.147 hostapd_radius_server Struct Reference

RADIUS server information for RADIUS client.

`#include <radius_client.h>`

Data Fields

- struct hostapd_ip_addr addr
  radiusAuthServerAddress or radiusAccServerAddress
- int port
  radiusAuthClientServerPortNumber or radiusAccClientServerPortNumber
- u8 * shared_secret
  Shared secret for authenticating RADIUS messages.
- size_t shared_secret_len
  Length of shared_secret in octets.
- int index
  radiusAuthServerIndex or radiusAccServerIndex
- int round_trip_time
  radiusAuthClientRoundTripTime or radiusAccClientRoundTripTime
- u32 requests
  radiusAuthClientAccessRequests or radiusAccClientRequests
- u32 retransmissions
  radiusAuthClientAccessRetransmissions or radiusAccClientRetransmissions
- u32 access_accepts
  radiusAuthClientAccessAccepts
- u32 access_rejects
  radiusAuthClientAccessRejects
- u32 access_challenges
  radiusAuthClientAccessChallenges
- u32 responses
  radiusAccClientResponses
- u32 malformed_responses
  radiusAuthClientMalformedAccessResponses or radiusAccClientMalformedResponses
- u32 bad_authenticators
  radiusAuthClientBadAuthenticators or radiusAccClientBadAuthenticators

- u32 timeouts

  radiusAuthClientTimeouts or radiusAccClientTimeouts

- u32 unknown_types

  radiusAuthClientUnknownTypes or radiusAccClientUnknownTypes

- u32 packets_dropped

  radiusAuthClientPacketsDropped or radiusAccClientPacketsDropped

14.147.1 Detailed Description

RADIUS server information for RADIUS client. This structure contains information about a RADIUS server. The values are mainly for MIB information. The MIB variable prefix (radiusAuth or radiusAcc) depends on whether this is an authentication or accounting server.

radiusAuthClientPendingRequests (or radiusAccClientPendingRequests) is the length of hapd->radius->msgs for matching msg_type.

14.147.2 Field Documentation

14.147.2.1 int hostapd_radius_server::round_trip_time

radiusAuthClientRoundTripTime or radiusAccClientRoundTripTime Round-trip time in hundredths of a second.

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

- src/radius/radius_client.h

14.148 hostapd_radius_servers Struct Reference

RADIUS servers for RADIUS client.

`#include <radius_client.h>`

Data Fields

- struct hostapd_radius_server * auth_servers
  RADIUS Authentication servers in priority order.
- int num_auth_servers
  Number of auth_servers entries.
- struct hostapd_radius_server * auth_server
  The current Authentication server.
- struct hostapd_radius_server * acct_servers
  RADIUS Accounting servers in priority order.
- int num_acct_servers
  Number of acct_servers entries.
- struct hostapd_radius_server * acct_server
  The current Accounting server.
- int retry_primary_interval
  Retry interval for trying primary server.
- int msg_dumps
  Whether RADIUS message details are shown in stdout.
- struct hostapd_ip_addr client_addr
  Client (local) address to use if force_client_addr.
- int force_client_addr
  Whether to force client (local) address.

14.148.1 Detailed Description

RADIUS servers for RADIUS client.

14.148.2 Field Documentation

14.148.2.1 int hostapd_radius_servers::retry_primary_interval

Retry interval for trying primary server. This specifies a retry interval in sexconds for trying to return to the primary RADIUS server. RADIUS client code will automatically try to use the next server when the

current server is not replying to requests. If this interval is set (non-zero), the primary server will be retried after the specified number of seconds has passed even if the current used secondary server is still working.

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

- src/radius/radius_client.h

14.149 hostapd_rate_data Struct Reference

Data Fields

- int **rate**
- int **flags**

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

- src/drivers/driver.h

14.150 hostapd_ssid Struct Reference

Data Fields

- char **ssid** [HOSTAPD_MAX_SSID_LEN+1]
- size_t **ssid_len**
- int **ssid_set**
- char **vlan** [IFNAMSIZ+1]
- secpolicy **security_policy**
- struct hostapd_wpa_psk * **wpa_psk**
- char * **wpa_passphrase**
- char * **wpa_psk_file**
- struct hostapd_wep_keys **wep**
- int **dynamic_vlan**
- struct hostapd_wep_keys ** **dyn_vlan_keys**
- size_t **max_dyn_vlan_keys**

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

- hostapd/config.h

14.151 hostapd_sta_add_params Struct Reference

Data Fields

- const u8 * **addr**
- u16 **aid**
- u16 **capability**
- const u8 * **supp_rates**
- size_t **supp_rates_len**
- int **flags**
- u16 **listen_interval**
- struct ht_cap_ie * **ht_capabilities**

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

- src/drivers/driver.h

14.152 hostapd_tx_queue_params Struct Reference

Data Fields

- int **aifs**
- int **cwmin**
- int **cwmax**
- int **burst**
- int **configured**

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

- hostapd/config.h

14.153 hostapd_vlan Struct Reference

Data Fields

- struct hostapd_vlan * **next**
- int **vlan_id**
- char **ifname** [IFNAMSIZ+1]
- int **dynamic_vlan**

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

- hostapd/config.h

14.154 hostapd_wep_keys Struct Reference

Data Fields

- u8 **idx**
- u8 * **key** [NUM_WEP_KEYS]
- size_t **len** [NUM_WEP_KEYS]
- int **keys_set**
- size_t **default_len**

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

- hostapd/config.h

14.155 hostapd_wmm_ac_params Struct Reference

Data Fields

- int **cwmin**
- int **cwmax**
- int **aifs**
- int **txop_limit**
- int **admission_control_mandatory**

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

- hostapd/config.h

14.156 hostapd_wpa_psk Struct Reference

Data Fields

- struct hostapd_wpa_psk * **next**
- int **group**
- u8 **psk** [PMK_LEN]
- u8 **addr** [ETH_ALEN]

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

- hostapd/config.h

14.157 ht_cap_ie Struct Reference

Data Fields

- u8 **id**
- u8 **length**
- struct ieee80211_ht_capability **data**

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

- src/common/ieee802_11_defs.h

14.158 http_client Struct Reference

Data Fields

- struct sockaddr_in **dst**
- int **sd**
- struct wpabuf * **req**
- size_t **req_pos**
- size_t **max_response**
- void(* **cb**)(void *ctx, struct http_client *c, enum http_client_event event)
- void * **cb_ctx**
- struct httpread * **hread**
- struct wpabuf **body**

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

- src/wps/http_client.c

14.159 http_request Struct Reference

Data Fields

- struct http_request * **next**
- struct http_server * **srv**
- int **fd**
- struct sockaddr_in **cli**
- struct httpread * **hread**

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

- src/wps/http_server.c

14.160 http_server Struct Reference

Data Fields

- void(∗ **cb**)(void ∗ctx, struct http_request ∗req)
- void ∗ **cb_ctx**
- int **fd**
- int **port**
- struct http_request ∗ **requests**
- unsigned int **request_count**

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

- src/wps/http_server.c

14.161 httpread Struct Reference

Public Types

- enum **trailer_state** { **trailer_line_begin** = 0, **trailer_empty_cr**, **trailer_nonempty**, **trailer_nonempty_cr** }

Data Fields

- int **sd**
- void(∗ **cb**)(struct httpread ∗handle, void ∗cookie, enum httpread_event e)
- void ∗ **cookie**
- int **max_bytes**
- int **timeout_seconds**
- int **sd_registered**
- int **to_registered**
- int **got_hdr**
- char **hdr** [HTTPREAD_HEADER_MAX_SIZE+1]
- int **hdr_nbytes**
- enum httpread_hdr_type **hdr_type**
- int **version**
- int **reply_code**
- int **got_content_length**
- int **content_length**
- int **chunked**
- char ∗ **uri**
- int **got_body**
- char ∗ **body**
- int **body_nbytes**
- int **body_alloc_nbytes**
- int **got_file**
- int **in_chunk_data**
- int **chunk_start**
- int **chunk_size**
- int **in_trailer**
- enum httpread::trailer_state **trailer_state**

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

- src/wps/httpread.c

14.162 i802_bss Struct Reference

Data Fields

- struct i802_bss * **next**
- int **ifindex**
- unsigned int **beacon_set**:1

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

- src/drivers/driver_nl80211.c

14.163 iapp_ack_security_block Struct Reference

Data Fields

- u8 **iv** [8]
- u8 **new_ap_ack_authenticator** [48]

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

- hostapd/iapp.c

14.164 iapp_add_notify Struct Reference

Data Fields

- u8 **addr_len**
- u8 **reserved**
- u8 **mac_addr** [ETH_ALEN]
- be16 **seq_num**

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

- hostapd/iapp.c

14.165 iapp_cache_notify Struct Reference

Data Fields

- u8 **addr_len**
- u8 **reserved**
- u8 **mac_addr** [ETH_ALEN]
- u16 **seq_num**
- u8 **current_ap** [ETH_ALEN]
- u16 **ctx_block_len**

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

- hostapd/iapp.c

14.166 iapp_cache_response Struct Reference

Data Fields

- u8 **addr_len**
- u8 **status**
- u8 **mac_addr** [ETH_ALEN]
- u16 **seq_num**

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

- hostapd/iapp.c

14.167 iapp_data Struct Reference

Data Fields

- struct hostapd_data * **hapd**
- u16 **identifier**
- struct in_addr own **multicast**
- int **udp_sock**
- int **packet_sock**

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

- hostapd/iapp.c

14.168 iapp_hdr Struct Reference

Data Fields

- u8 **version**
- u8 **command**
- be16 **identifier**
- be16 **length**

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

- hostapd/iapp.c

14.169 iapp_layer2_update Struct Reference

Data Fields

- u8 **da** [ETH_ALEN]
- u8 **sa** [ETH_ALEN]
- be16 **len**
- u8 **dsap**
- u8 **ssap**
- u8 **control**
- u8 **xid_info** [3]

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

- hostapd/iapp.c

14.170 iapp_move_notify Struct Reference

Data Fields

- u8 **addr_len**
- u8 **reserved**
- u8 **mac_addr** [ETH_ALEN]
- u16 **seq_num**
- u16 **ctx_block_len**

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

- hostapd/iapp.c

14.171 iapp_move_response Struct Reference

Data Fields

- u8 **addr_len**
- u8 **status**
- u8 **mac_addr** [ETH_ALEN]
- u16 **seq_num**
- u16 **ctx_block_len**

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

- hostapd/iapp.c

14.172 iapp_send_security_block Struct Reference

Data Fields

- u8 **iv** [8]
- u16 **sec_block_len**

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

- hostapd/iapp.c

14.173 ibss_rsn Struct Reference

Data Fields

- struct wpa_supplicant * **wpa_s**
- struct wpa_authenticator * **auth_group**
- struct ibss_rsn_peer * **peers**
- u8 **psk** [PMK_LEN]

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

- wpa_supplicant/ibss_rsn.h

14.174 ibss_rsn_peer Struct Reference

Data Fields

- struct ibss_rsn_peer * **next**
- struct ibss_rsn * **ibss_rsn**
- u8 **addr** [ETH_ALEN]
- struct wpa_sm * **supp**
- wpa_states **supp_state**
- u8 **supp_ie** [80]
- size_t **supp_ie_len**
- struct wpa_state_machine * **auth**

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

- wpa_supplicant/ibss_rsn.h

14.175 wpa_event_data::ibss_rsn_start Struct Reference

Data for EVENT_IBSS_RSN_START.

`#include <driver.h>`

Data Fields

- u8 **peer** [ETH_ALEN]

14.175.1 Detailed Description

Data for EVENT_IBSS_RSN_START.

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

- src/drivers/driver.h

14.176 ieee80211_frame_info Struct Reference

Data Fields

- u32 **version**
- u32 **length**
- u64 **mactime**
- u64 **hosttime**
- u32 **phytype**
- u32 **channel**
- u32 **datarate**
- u32 **antenna**
- u32 **priority**
- u32 **ssi_type**
- u32 **ssi_signal**
- u32 **ssi_noise**
- u32 **preamble**
- u32 **encoding**
- u32 **msg_type**

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

- hostapd/ap_list.c

14.177 ieee80211_hdr Struct Reference

Data Fields

- le16 **frame_control**
- le16 **duration_id**
- u8 **addr1** [6]
- u8 **addr2** [6]
- u8 **addr3** [6]
- le16 **seq_ctrl**

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

- src/common/ieee802_11_defs.h

14.178 ieee80211_ht_capability Struct Reference

Data Fields

- le16 **capabilities_info**
- u8 **mac_ht_params_info**
- u8 **supported_mcs_set** [16]
- le16 **extended_ht_capability_info**
- le32 **tx_BF_capability_info**
- u8 **antenna_selection_info**

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

- src/common/ieee802_11_defs.h

14.179 ieee80211_ht_operation Struct Reference

Data Fields

- u8 **control_chan**
- u8 **ht_param**
- le16 **operation_mode**
- le16 **stbc_param**
- u8 **basic_set** [16]

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

- src/common/ieee802_11_defs.h

14.180 ieee80211_mgmt Struct Reference

Data Fields

- le16 **frame_control**
- le16 **duration**
- u8 **da** [6]
- u8 **sa** [6]
- u8 **bssid** [6]
- le16 **seq_ctrl**
- union {
  struct {
  le16 **auth_alg**
  le16 **auth_transaction**
  le16 **status_code**
  u8 **variable** [0]
  } **auth**
  struct {
  le16 **reason_code**
  } **deauth**
  struct {
  le16 **capab_info**
  le16 **listen_interval**
  u8 **variable** [0]
  } **assoc_req**
  struct {
  le16 **capab_info**
  le16 **status_code**
  le16 **aid**
  u8 **variable** [0]
  } **assoc_resp**
  struct {
  le16 **capab_info**
  le16 **status_code**
  le16 **aid**
  u8 **variable** [0]
  } **reassoc_resp**
  struct {
  le16 **capab_info**
  le16 **listen_interval**
  u8 **current_ap** [6]
  u8 **variable** [0]
  } **reassoc_req**
  struct {
  le16 **reason_code**
  } **disassoc**
  struct {
  u8 **timestamp** [8]
  le16 **beacon_int**
  le16 **capab_info**
  u8 **variable** [0]
  } **beacon**
  struct {

```
      u8 variable [0]
    } probe_req
    struct {
      u8 timestamp [8]
      le16 beacon_int
      le16 capab_info
      u8 variable [0]
    } probe_resp
    struct {
      u8 category
      union {
        struct {
          u8 action_code
          u8 dialog_token
          u8 status_code
          u8 variable [0]
        } wmm_action
        struct {
          u8 action_code
          u8 element_id
          u8 length
          u8 switch_mode
          u8 new_chan
          u8 switch_count
        } chan_switch
        struct {
          u8 action
          u8 sta_addr [ETH_ALEN]
          u8 target_ap_addr [ETH_ALEN]
          u8 variable [0]
        } ft_action_req
        struct {
          u8 action
          u8 sta_addr [ETH_ALEN]
          u8 target_ap_addr [ETH_ALEN]
          le16 status_code
          u8 variable [0]
        } ft_action_resp
        struct {
          u8 action
          u8 trans_id [WLAN_SA_QUERY_TR_ID_LEN]
        } sa_query_req
        struct {
          u8 action
          u8 trans_id [WLAN_SA_QUERY_TR_ID_LEN]
        } sa_query_resp
      } u
    } action
  } u
```

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

- src/common/ieee802_11_defs.h

14.181 ieee80211_radiotap_header Struct Reference

Data Fields

- uint8_t **it_version**
- uint8_t **it_pad**
- uint16_t **it_len**
- uint32_t **it_present**

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

- src/utils/radiotap.h

14.182 ieee80211_radiotap_iterator Struct Reference

tracks walk thru present radiotap args

`#include <radiotap_iter.h>`

Data Fields

- struct ieee80211_radiotap_header * **rtheader**
- int **max_length**
- int **this_arg_index**
- unsigned char * **this_arg**
- int **arg_index**
- unsigned char * **arg**
- uint32_t * **next_bitmap**
- uint32_t **bitmap_shifter**

14.182.1 Detailed Description

tracks walk thru present radiotap args

Parameters:

rtheader pointer to the radiotap header we are walking through

max_length length of radiotap header in cpu byte ordering

this_arg_index IEEE80211_RADIOTAP_... index of current arg

this_arg pointer to current radiotap arg

arg_index internal next argument index

arg internal next argument pointer

next_bitmap internal pointer to next present u32

bitmap_shifter internal shifter for curr u32 bitmap, b0 set == arg present

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

- src/utils/radiotap_iter.h

14.183 ieee80211_rx_status Struct Reference

Data Fields

- int **channel**
- int **ssi**

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

- src/drivers/driver.h

14.184 ieee80211_sta_bss Struct Reference

Data Fields

- struct ieee80211_sta_bss * **next**
- struct ieee80211_sta_bss * **hnext**
- u8 **bssid** [ETH_ALEN]
- u8 **ssid** [MAX_SSID_LEN]
- size_t **ssid_len**
- u16 **capability**
- int **hw_mode**
- int **channel**
- int **freq**
- int **rssi**
- u8 * **ie**
- size_t **ie_len**
- u8 * **wpa_ie**
- size_t **wpa_ie_len**
- u8 * **rsn_ie**
- size_t **rsn_ie_len**
- u8 * **wmm_ie**
- size_t **wmm_ie_len**
- u8 * **mdie**
- size_t **mdie_len**
- u8 **supp_rates** [IEEE80211_MAX_SUPP_RATES]
- size_t **supp_rates_len**
- int **beacon_int**
- u64 **timestamp**
- int **probe_resp**
- struct os_time **last_update**

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

- wpa_supplicant/mlme.c

14.185 ieee8023_hdr Struct Reference

Data Fields

- u8 **dest** [6]
- u8 **src** [6]
- u16 **ethertype**

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

- src/drivers/driver_wired.c

14.186 ieee802_11_elems Struct Reference

Data Fields

- u8 * **ssid**
- u8 **ssid_len**
- u8 * **supp_rates**
- u8 **supp_rates_len**
- u8 * **fh_params**
- u8 **fh_params_len**
- u8 * **ds_params**
- u8 **ds_params_len**
- u8 * **cf_params**
- u8 **cf_params_len**
- u8 * **tim**
- u8 **tim_len**
- u8 * **ibss_params**
- u8 **ibss_params_len**
- u8 * **challenge**
- u8 **challenge_len**
- u8 * **erp_info**
- u8 **erp_info_len**
- u8 * **ext_supp_rates**
- u8 **ext_supp_rates_len**
- u8 * **wpa_ie**
- u8 **wpa_ie_len**
- u8 * **rsn_ie**
- u8 **rsn_ie_len**
- u8 * **wmm**
- u8 **wmm_len**
- u8 * **wmm_tspec**
- u8 **wmm_tspec_len**
- u8 * **wps_ie**
- u8 **wps_ie_len**
- u8 * **power_cap**
- u8 **power_cap_len**
- u8 * **supp_channels**
- u8 **supp_channels_len**
- u8 * **mdie**
- u8 **mdie_len**
- u8 * **ftie**
- u8 **ftie_len**
- u8 * **timeout_int**
- u8 **timeout_int_len**
- u8 * **ht_capabilities**
- u8 **ht_capabilities_len**
- u8 * **ht_operation**
- u8 **ht_operation_len**
- u8 * **vendor_ht_cap**
- u8 **vendor_ht_cap_len**

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

- src/common/ieee802_11_common.h

14.187 ieee802_1x_eapol_key Struct Reference

Data Fields

- u8 **type**
- u16 **key_length**
- u8 **replay_counter** [8]
- u8 **key_iv** [16]
- u8 **key_index**
- u8 **key_signature** [16]
- u8 **key_length** [2]

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

- hostapd/ieee802_1x.h
- src/eapol_supp/eapol_supp_sm.c

14.188 ieee802_1x_hdr Struct Reference

Data Fields

- u8 **version**
- u8 **type**
- be16 **length**

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

- src/common/eapol_common.h

14.189 ieee802_3_hdr_s Struct Reference

Data Fields

- unsigned char **da** [6]
- unsigned char **sa** [6]
- unsigned short **type**

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

- src/drivers/prism54.h

14.190 ifinfomsg Struct Reference

Data Fields

- unsigned char **ifi_family**
- unsigned char **__ifi_pad**
- unsigned short **ifi_type**
- int **ifi_index**
- unsigned **ifi_flags**
- unsigned **ifi_change**

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

- src/drivers/priv_netlink.h

14.191 ikev2_encr_alg Struct Reference

Data Fields

- int **id**
- size_t **key_len**
- size_t **block_size**

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

- src/eap_common/ikev2_common.h

14.192 ikev2_hdr Struct Reference

Data Fields

- u8 **i_spi** [IKEV2_SPI_LEN]
- u8 **r_spi** [IKEV2_SPI_LEN]
- u8 **next_payload**
- u8 **version**
- u8 **exchange_type**
- u8 **flags**
- u8 **message_id** [4]
- u8 **length** [4]

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

- src/eap_common/ikev2_common.h

14.193 ikev2_initiator_data Struct Reference

Public Types

- enum { **SA_INIT**, **SA_AUTH**, **CHILD_SA**, **IKEV2_DONE** }
- enum { **PEER_AUTH_CERT**, **PEER_AUTH_SECRET** }

Data Fields

- enum ikev2_initiator_data:: { ... } **state**
- u8 **i_spi** [IKEV2_SPI_LEN]
- u8 **r_spi** [IKEV2_SPI_LEN]
- u8 **i_nonce** [IKEV2_NONCE_MAX_LEN]
- size_t **i_nonce_len**
- u8 **r_nonce** [IKEV2_NONCE_MAX_LEN]
- size_t **r_nonce_len**
- struct wpabuf * **r_dh_public**
- struct wpabuf * **i_dh_private**
- struct ikev2_proposal_data **proposal**
- struct dh_group * **dh**
- struct ikev2_keys **keys**
- u8 * **IDi**
- size_t **IDi_len**
- u8 * **IDr**
- size_t **IDr_len**
- u8 **IDr_type**
- struct wpabuf * **r_sign_msg**
- struct wpabuf * **i_sign_msg**
- u8 * **shared_secret**
- size_t **shared_secret_len**
- enum ikev2_initiator_data:: { ... } **peer_auth**
- u8 * **key_pad**
- size_t **key_pad_len**
- const u8 *(* **get_shared_secret**)(void *ctx, const u8 *IDr, size_t IDr_len, size_t *secret_len)
- void * **cb_ctx**
- int **unknown_user**

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

- src/eap_server/ikev2.h

14.194 ikev2_integ_alg Struct Reference

Data Fields

- int **id**
- size_t **key_len**
- size_t **hash_len**

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

- src/eap_common/ikev2_common.h

14.195 ikev2_keys Struct Reference

Data Fields

- u8 * **SK_d**
- u8 * **SK_ai**
- u8 * **SK_ar**
- u8 * **SK_ei**
- u8 * **SK_er**
- u8 * **SK_pi**
- u8 * **SK_pr**
- size_t **SK_d_len**
- size_t **SK_integ_len**
- size_t **SK_encr_len**
- size_t **SK_prf_len**

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

- src/eap_common/ikev2_common.h

14.196 ikev2_payload_hdr Struct Reference

Data Fields

- u8 **next_payload**
- u8 **flags**
- u8 **payload_length** [2]

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

- src/eap_common/ikev2_common.h

14.197 ikev2_payloads Struct Reference

Data Fields

- const u8 * **sa**
- size_t **sa_len**
- const u8 * **ke**
- size_t **ke_len**
- const u8 * **idi**
- size_t **idi_len**
- const u8 * **idr**
- size_t **idr_len**
- const u8 * **cert**
- size_t **cert_len**
- const u8 * **auth**
- size_t **auth_len**
- const u8 * **nonce**
- size_t **nonce_len**
- const u8 * **encrypted**
- size_t **encrypted_len**
- u8 **encr_next_payload**
- const u8 * **notification**
- size_t **notification_len**

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

- src/eap_common/ikev2_common.h

14.198 ikev2_prf_alg Struct Reference

Data Fields

- int **id**
- size_t **key_len**
- size_t **hash_len**

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

- src/eap_common/ikev2_common.h

14.199 ikev2_proposal Struct Reference

Data Fields

- u8 **type**
- u8 **reserved**
- u8 **proposal_length** [2]
- u8 **proposal_num**
- u8 **protocol_id**
- u8 **spi_size**
- u8 **num_transforms**

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

- src/eap_common/ikev2_common.h

14.200 ikev2_proposal_data Struct Reference

Data Fields

- u8 **proposal_num**
- int **integ**
- int **prf**
- int **encr**
- int **dh**

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

- src/eap_peer/ikev2.h
- src/eap_server/ikev2.h

14.201 ikev2_responder_data Struct Reference

Public Types

- enum {
  SA_INIT, **SA_AUTH**, **CHILD_SA**, **NOTIFY**,
  IKEV2_DONE, **IKEV2_FAILED** }
- enum { **PEER_AUTH_CERT**, **PEER_AUTH_SECRET** }
- enum { **LAST_MSG_SA_INIT**, **LAST_MSG_SA_AUTH** }

Data Fields

- enum ikev2_responder_data:: { ... } **state**
- u8 **i_spi** [IKEV2_SPI_LEN]
- u8 **r_spi** [IKEV2_SPI_LEN]
- u8 **i_nonce** [IKEV2_NONCE_MAX_LEN]
- size_t **i_nonce_len**
- u8 **r_nonce** [IKEV2_NONCE_MAX_LEN]
- size_t **r_nonce_len**
- struct wpabuf * **i_dh_public**
- struct wpabuf * **r_dh_private**
- struct ikev2_proposal_data **proposal**
- struct dh_group * **dh**
- struct ikev2_keys **keys**
- u8 * **IDi**
- size_t **IDi_len**
- u8 **IDi_type**
- u8 * **IDr**
- size_t **IDr_len**
- struct wpabuf * **r_sign_msg**
- struct wpabuf * **i_sign_msg**
- u8 * **shared_secret**
- size_t **shared_secret_len**
- enum ikev2_responder_data:: { ... } **peer_auth**
- u8 * **key_pad**
- size_t **key_pad_len**
- u16 **error_type**
- enum ikev2_responder_data:: { ... } **last_msg**

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

- src/eap_peer/ikev2.h

14.202 ikev2_transform Struct Reference

Data Fields

- u8 **type**
- u8 **reserved**
- u8 **transform_length** [2]
- u8 **transform_type**
- u8 **reserved2**
- u8 **transform_id** [2]

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

- src/eap_common/ikev2_common.h

14.203 wpa_event_data::interface_status Struct Reference

Data for EVENT_INTERFACE_STATUS.

`#include <driver.h>`

Public Types

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

Data Fields

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

14.203.1 Detailed Description

Data for EVENT_INTERFACE_STATUS.

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

- src/drivers/driver.h

14.204 ipw_param Struct Reference

Data Fields

- u32 **cmd**
- u8 **sta_addr** [ETH_ALEN]
- union {
  struct {
  u8 **name**
  u32 **value**
  } **wpa_param**
  struct {
  u32 **len**
  u8 **reserved** [32]
  u8 **data** [0]
  } **wpa_ie**
  struct {
  u32 **command**
  u32 **reason_code**
  } **mlme**
  struct {
  u8 **alg** [IPW_CRYPT_ALG_NAME_LEN]
  u8 **set_tx**
  u32 **err**
  u8 **idx**
  u8 **seq** [8]
  u16 **key_len**
  u8 **key** [0]
  } **crypt**
  } **u**

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

- src/drivers/driver_ipw.c

14.205 iw_discarded Struct Reference

Data Fields

- __u32 **nwid**
- __u32 **code**
- __u32 **fragment**
- __u32 **retries**
- __u32 **misc**

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

- src/common/wireless_copy.h

14.206 iw_encode_ext Struct Reference

Data Fields

- __u32 **ext_flags**
- __u8 **tx_seq** [IW_ENCODE_SEQ_MAX_SIZE]
- __u8 **rx_seq** [IW_ENCODE_SEQ_MAX_SIZE]
- struct sockaddr **addr**
- __u16 **alg**
- __u16 **key_len**
- __u8 **key** [0]

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

- src/common/wireless_copy.h

14.207 iw_event Struct Reference

Data Fields

- __u16 **len**
- __u16 **cmd**
- union iwreq_data **u**

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

- src/common/wireless_copy.h

14.208 iw_freq Struct Reference

Data Fields

- __s32 **m**
- __s16 **e**
- __u8 **i**
- __u8 **flags**

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

- src/common/wireless_copy.h

14.209 iw_michaelmicfailure Struct Reference

Data Fields

- __u32 **flags**
- struct sockaddr **src_addr**
- __u8 **tsc** [IW_ENCODE_SEQ_MAX_SIZE]

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

- src/common/wireless_copy.h

14.210 iw_missed Struct Reference

Data Fields

- __u32 **beacon**

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

- src/common/wireless_copy.h

14.211 iw_mlme Struct Reference

Data Fields

- __u16 **cmd**
- __u16 **reason_code**
- struct sockaddr **addr**

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

- src/common/wireless_copy.h

14.212 iw_param Struct Reference

Data Fields

- __s32 **value**
- __u8 **fixed**
- __u8 **disabled**
- __u16 **flags**

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

- src/common/wireless_copy.h

14.213 iw_pmkid_cand Struct Reference

Data Fields

- __u32 **flags**
- __u32 **index**
- struct sockaddr **bssid**

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

- src/common/wireless_copy.h

14.214 iw_pmksa Struct Reference

Data Fields

- __u32 **cmd**
- struct sockaddr **bssid**
- __u8 **pmkid** [IW_PMKID_LEN]

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

- src/common/wireless_copy.h

14.215 iw_point Struct Reference

Data Fields

- void __user * **pointer**
- __u16 **length**
- __u16 **flags**

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

- src/common/wireless_copy.h

14.216 iw_priv_args Struct Reference

Data Fields

- __u32 **cmd**
- __u16 **set_args**
- __u16 **get_args**
- char **name** [IFNAMSIZ]

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

- src/common/wireless_copy.h

14.217 iw_quality Struct Reference

Data Fields

- __u8 **qual**
- __u8 **level**
- __u8 **noise**
- __u8 **updated**

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

- src/common/wireless_copy.h

14.218 iw_range Struct Reference

Data Fields

- __u32 **throughput**
- __u32 **min_nwid**
- __u32 **max_nwid**
- __u16 **old_num_channels**
- __u8 **old_num_frequency**
- __u32 **event_capa** [6]
- __s32 **sensitivity**
- struct iw_quality **max_qual**
- struct iw_quality **avg_qual**
- __u8 **num_bitrates**
- __s32 **bitrate** [IW_MAX_BITRATES]
- __s32 **min_rts**
- __s32 **max_rts**
- __s32 **min_frag**
- __s32 **max_frag**
- __s32 **min_pmp**
- __s32 **max_pmp**
- __s32 **min_pmt**
- __s32 **max_pmt**
- __u16 **pmp_flags**
- __u16 **pmt_flags**
- __u16 **pm_capa**
- __u16 **encoding_size** [IW_MAX_ENCODING_SIZES]
- __u8 **num_encoding_sizes**
- __u8 **max_encoding_tokens**
- __u8 **encoding_login_index**
- __u16 **txpower_capa**
- __u8 **num_txpower**
- __s32 **txpower** [IW_MAX_TXPOWER]
- __u8 **we_version_compiled**
- __u8 **we_version_source**
- __u16 **retry_capa**
- __u16 **retry_flags**
- __u16 **r_time_flags**
- __s32 **min_retry**
- __s32 **max_retry**
- __s32 **min_r_time**
- __s32 **max_r_time**
- __u16 **num_channels**
- __u8 **num_frequency**
- struct iw_freq **freq** [IW_MAX_FREQUENCIES]
- __u32 **enc_capa**

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

- src/common/wireless_copy.h

14.219 iw_scan_req Struct Reference

Data Fields

- __u8 **scan_type**
- __u8 **essid_len**
- __u8 **num_channels**
- __u8 **flags**
- struct sockaddr **bssid**
- __u8 **essid** [IW_ESSID_MAX_SIZE]
- __u32 **min_channel_time**
- __u32 **max_channel_time**
- struct iw_freq **channel_list** [IW_MAX_FREQUENCIES]

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

- src/common/wireless_copy.h

14.220 iw_statistics Struct Reference

Data Fields

- __u16 **status**
- struct iw_quality **qual**
- struct iw_discarded **discard**
- struct iw_missed **miss**

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

- src/common/wireless_copy.h

14.221 iw_thrspy Struct Reference

Data Fields

- struct sockaddr **addr**
- struct iw_quality **qual**
- struct iw_quality **low**
- struct iw_quality **high**

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

- src/common/wireless_copy.h

14.222 iwreq Struct Reference

Data Fields

- union {
  char **ifrn_name** [IFNAMSIZ]
  } **ifr_ifrn**

- union iwreq_data **u**

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

- src/common/wireless_copy.h

14.223 iwreq_data Union Reference

Data Fields

- char **name** [IFNAMSIZ]
- struct iw_point **essid**
- struct iw_param **nwid**
- struct iw_freq **freq**
- struct iw_param **sens**
- struct iw_param **bitrate**
- struct iw_param **txpower**
- struct iw_param **rts**
- struct iw_param **frag**
- __u32 **mode**
- struct iw_param **retry**
- struct iw_point **encoding**
- struct iw_param **power**
- struct iw_quality **qual**
- struct sockaddr **ap_addr**
- struct sockaddr **addr**
- struct iw_param **param**
- struct iw_point **data**

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

- src/common/wireless_copy.h

14.224 l2_ethhdr Struct Reference

Data Fields

- u8 **h_dest** [ETH_ALEN]
- u8 **h_source** [ETH_ALEN]
- be16 **h_proto**

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

- src/l2_packet/l2_packet.h

14.225 l2_packet_data Struct Reference

Data Fields

- pcap_t * **pcap**
- char **ifname** [100]
- u8 **own_addr** [ETH_ALEN]
- void(* **rx_callback**)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len)
- void * **rx_callback_ctx**
- int **l2_hdr**
- int **fd**
- int **ifindex**
- HANDLE **rx_avail**
- OVERLAPPED **rx_overlapped**
- u8 **rx_buf** [1514]
- DWORD **rx_written**
- eth_t * **eth**
- char * **own_socket_path**
- struct sockaddr_un **priv_addr**
- unsigned int **num_fast_poll**
- int **running**
- HANDLE **rx_done**
- HANDLE **rx_thread**
- HANDLE **rx_thread_done**
- HANDLE **rx_notify**
- u8 * **rx_buf**
- u8 * **rx_src**
- size_t **rx_len**
- size_t **rx_no_wait**

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

- src/l2_packet/l2_packet_freebsd.c
- src/l2_packet/l2_packet_linux.c
- src/l2_packet/l2_packet_ndis.c
- src/l2_packet/l2_packet_none.c
- src/l2_packet/l2_packet_pcap.c
- src/l2_packet/l2_packet_privsep.c
- src/l2_packet/l2_packet_winpcap.c

14.226 l2_packet_ndisuio_global Struct Reference

Data Fields

- int **refcount**
- unsigned short **first_proto**
- struct l2_packet_data * **l2** [2]

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

- src/l2_packet/l2_packet_ndis.c

14.227 mac_acl_entry Struct Reference

Data Fields

- macaddr **addr**
- int **vlan_id**

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

- hostapd/config.h

14.228 madwifi_driver_data Struct Reference

Data Fields

- struct hostapd_data * **hapd**
- char **iface** [IFNAMSIZ+1]
- int **ifindex**
- struct l2_packet_data * **sock_xmit**
- struct l2_packet_data * **sock_recv**
- int **ioctl_sock**
- int **wext_sock**
- int **we_version**
- u8 **acct_mac** [ETH_ALEN]
- struct hostap_sta_driver_data **acct_data**
- struct l2_packet_data * **sock_raw**

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

- src/drivers/driver_atheros.c

14.229 MD4Context Struct Reference

Data Fields

- u32 **state** [4]
- u64 **count**
- u8 **buffer** [MD4_BLOCK_LENGTH]

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

- src/crypto/md4-internal.c

14.230 MD5Context Struct Reference

Data Fields

- u32 **buf** [4]
- u32 **bits** [2]
- u8 **in** [64]

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

- src/crypto/md5_i.h

14.231 wpa_event_data::michael_mic_failure Struct Reference

Data for EVENT_MICHAEL_MIC_FAILURE.

`#include <driver.h>`

Data Fields

- int **unicast**
- const u8 * **src**

14.231.1 Detailed Description

Data for EVENT_MICHAEL_MIC_FAILURE.

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

- src/drivers/driver.h

14.232 milenage_parameters Struct Reference

Data Fields

- struct milenage_parameters * **next**
- char **imsi** [20]
- u8 **ki** [16]
- u8 **opc** [16]
- u8 **amf** [2]
- u8 **sqn** [6]

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

- src/hlr_auc_gw/hlr_auc_gw.c

14.233 mimo_pwr_save_action Struct Reference

Data Fields

- u8 **category**
- u8 **action**
- u8 **enable**
- u8 **mode**

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

- src/common/ieee802_11_defs.h

14.234 mp_int Struct Reference

Data Fields

- int **used**
- int **alloc**
- int **sign**
- mp_digit * **dp**

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

- src/tls/libtommath.c

14.235 ms_change_password Struct Reference

Data Fields

- u8 **encr_password** [516]
- u8 **encr_hash** [16]
- u8 **peer_challenge** [MSCHAPV2_CHAL_LEN]
- u8 **reserved** [8]
- u8 **nt_response** [MSCHAPV2_NT_RESPONSE_LEN]
- u8 **flags** [2]

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

- src/eap_peer/eap_mschapv2.c

14.236 ms_response Struct Reference

Data Fields

- u8 **peer_challenge** [MSCHAPV2_CHAL_LEN]
- u8 **reserved** [8]
- u8 **nt_response** [MSCHAPV2_NT_RESPONSE_LEN]
- u8 **flags**

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

- src/eap_peer/eap_mschapv2.c

14.237 ndef_record Struct Reference

Data Fields

- u8 * **type**
- u8 * **id**
- u8 * **payload**
- u8 **type_length**
- u8 **id_length**
- u32 **payload_length**
- u32 **total_length**

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

- src/wps/ndef.c

14.238 NDIS_802_11_AI_REQFI Struct Reference

Data Fields

- USHORT **Capabilities**
- USHORT **ListenInterval**
- NDIS_802_11_MAC_ADDRESS **CurrentAPAddress**

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

- src/drivers/driver_ndis.c

14.239 NDIS_802_11_AI_RESFI Struct Reference

Data Fields

- USHORT **Capabilities**
- USHORT **StatusCode**
- USHORT **AssociationId**

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

- src/drivers/driver_ndis.c

14.240 NDIS_802_11_ASSOCIATION_INFORMATION Struct Reference

Data Fields

- ULONG **Length**
- USHORT **AvailableRequestFixedIEs**
- NDIS_802_11_AI_REQFI **RequestFixedIEs**
- ULONG **RequestIELength**
- ULONG **OffsetRequestIEs**
- USHORT **AvailableResponseFixedIEs**
- NDIS_802_11_AI_RESFI **ResponseFixedIEs**
- ULONG **ResponseIELength**
- ULONG **OffsetResponseIEs**

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

- src/drivers/driver_ndis.c

14.241 NDIS_802_11_AUTHENTICATION_ENCRYPTION Struct Reference

Data Fields

- NDIS_802_11_AUTHENTICATION_MODE **AuthModeSupported**
- NDIS_802_11_ENCRYPTION_STATUS **EncryptStatusSupported**

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

- src/drivers/driver_ndis.c

14.242 NDIS_802_11_AUTHENTICATION_REQUEST Struct Reference

Data Fields

- ULONG **Length**
- NDIS_802_11_MAC_ADDRESS **Bssid**
- ULONG **Flags**

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

- src/drivers/driver_ndis.c

14.243 NDIS_802_11_BSSID_LIST_EX Struct Reference

Data Fields

- ULONG **NumberOfItems**
- NDIS_WLAN_BSSID_EX **Bssid** [1]

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

- src/drivers/driver_ndis.c

14.244 NDIS_802_11_CAPABILITY Struct Reference

Data Fields

- ULONG **Length**
- ULONG **Version**
- ULONG **NoOfPMKIDs**
- ULONG **NoOfAuthEncryptPairsSupported**
- NDIS_802_11_AUTHENTICATION_ENCRYPTION **AuthenticationEncryptionSupported** [1]

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

- src/drivers/driver_ndis.c

14.245 NDIS_802_11_CONFIGURATION Struct Reference

Data Fields

- ULONG **Length**
- ULONG **BeaconPeriod**
- ULONG **ATIMWindow**
- ULONG **DSConfig**
- NDIS_802_11_CONFIGURATION_FH **FHConfig**

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

- src/drivers/driver_ndis.c

14.246 NDIS_802_11_CONFIGURATION_FH Struct Reference

Data Fields

- ULONG **Length**
- ULONG **HopPattern**
- ULONG **HopSet**
- ULONG **DwellTime**

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

- src/drivers/driver_ndis.c

14.247 NDIS_802_11_FIXED_IEs Struct Reference

Data Fields

- UCHAR **Timestamp** [8]
- USHORT **BeaconInterval**
- USHORT **Capabilities**

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

- src/drivers/driver_ndis.c

14.248 NDIS_802_11_KEY Struct Reference

Data Fields

- ULONG **Length**
- ULONG **KeyIndex**
- ULONG **KeyLength**
- NDIS_802_11_MAC_ADDRESS **BSSID**
- NDIS_802_11_KEY_RSC **KeyRSC**
- UCHAR **KeyMaterial** [1]

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

- src/drivers/driver_ndis.c

14.249 NDIS_802_11_PMKID Struct Reference

Data Fields

- ULONG **Length**
- ULONG **BSSIDInfoCount**
- BSSID_INFO **BSSIDInfo** [1]

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

- src/drivers/driver_ndis.c

14.250 NDIS_802_11_PMKID_CANDIDATE_LIST Struct Reference

Data Fields

- ULONG **Version**
- ULONG **NumCandidates**
- PMKID_CANDIDATE **CandidateList** [1]

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

- src/drivers/driver_ndis.c

14.251 NDIS_802_11_REMOVE_KEY Struct Reference

Data Fields

- ULONG **Length**
- ULONG **KeyIndex**
- NDIS_802_11_MAC_ADDRESS **BSSID**

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

- src/drivers/driver_ndis.c

14.252 NDIS_802_11_SSID Struct Reference

Data Fields

- ULONG **SsidLength**
- UCHAR **Ssid** [NDIS_802_11_LENGTH_SSID]

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

- src/drivers/driver_ndis.c

14.253 NDIS_802_11_STATUS_INDICATION Struct Reference

Data Fields

- NDIS_802_11_STATUS_TYPE **StatusType**

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

- src/drivers/driver_ndis.c

14.254 NDIS_802_11_WEP Struct Reference

Data Fields

- ULONG **Length**
- ULONG **KeyIndex**
- ULONG **KeyLength**
- UCHAR **KeyMaterial** [1]

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

- src/drivers/driver_ndis.c

14.255 ndis_events_data Struct Reference

Data Fields

- IWbemObjectSink **sink**
- IWbemObjectSinkVtbl **sink_vtbl**
- IWbemServices * **pSvc**
- IWbemLocator * **pLoc**
- HANDLE **read_pipe**
- HANDLE **write_pipe**
- HANDLE **event_avail**
- UINT **ref**
- int **terminating**
- char * **ifname**
- WCHAR * **adapter_desc**

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

- src/drivers/ndis_events.c

14.256 ndis_pmkid_entry Struct Reference

Data Fields

- struct ndis_pmkid_entry * **next**
- u8 **bssid** [ETH_ALEN]
- u8 **pmkid** [16]

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

- src/drivers/driver_ndis.h
- src/drivers/driver_ralink.h

14.257 NDIS_WLAN_BSSID_EX Struct Reference

Data Fields

- ULONG **Length**
- NDIS_802_11_MAC_ADDRESS **MacAddress**
- UCHAR **Reserved** [2]
- NDIS_802_11_SSID **Ssid**
- ULONG **Privacy**
- NDIS_802_11_RSSI **Rssi**
- NDIS_802_11_NETWORK_TYPE **NetworkTypeInUse**
- NDIS_802_11_CONFIGURATION **Configuration**
- NDIS_802_11_NETWORK_INFRASTRUCTURE **InfrastructureMode**
- NDIS_802_11_RATES_EX **SupportedRates**
- ULONG **IELength**
- UCHAR **IEs** [1]

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

- src/drivers/driver_ndis.c

14.258 network_handler_args Struct Reference

Data Fields

- struct wpa_supplicant * **wpa_s**
- struct wpa_ssid * **ssid**

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

- wpa_supplicant/ctrl_iface_dbus_new_handlers.h

14.259 nl80211_sta_flag_update Struct Reference

station flags mask/set

`#include <nl80211_copy.h>`

Data Fields

- __u32 **mask**
- __u32 **set**

14.259.1 Detailed Description

station flags mask/set

Parameters:

mask mask of station flags to set

set which values to set them to

Both mask and set contain bits as per &enum nl80211_sta_flags.

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

- src/common/nl80211_copy.h

14.260 nlmsghdr Struct Reference

Data Fields

- u32 **nlmsg_len**
- u16 **nlmsg_type**
- u16 **nlmsg_flags**
- u32 **nlmsg_seq**
- u32 **nlmsg_pid**

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

- src/drivers/priv_netlink.h

14.261 none_driver_data Struct Reference

Data Fields

- struct hostapd_data * **hapd**
- void * **ctx**

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

- src/drivers/driver_none.c

14.262 obj_attachment Struct Reference

Data Fields

- char **type**
- char **reserved**
- short **id**
- short **size**
- char **data** [1]

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

- src/drivers/prism54.h

14.263 obj_attachment_hdr Struct Reference

Data Fields

- char **type**
- char **reserved**
- short **id**
- short **size**

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

- src/drivers/prism54.h

14.264 obj_key Struct Reference

Data Fields

- char **type**
- char **length**
- char **key** [32]

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

- src/drivers/prism54.h

14.265 obj_mlme Struct Reference

Data Fields

- char **address** [6]
- short **id**
- short **state**
- short **code**

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

- src/drivers/prism54.h

14.266 obj_mlmeex Struct Reference

Data Fields

- char **address** [6]
- short **id**
- short **state**
- short **code**
- short **size**
- char **data** [1]

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

- src/drivers/prism54.h

14.267 obj_ssid Struct Reference

Data Fields

- char **length**
- char **octets** [33]

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

- src/drivers/prism54.h

14.268 obj_sta Struct Reference

Data Fields

- char **address** [6]
- char **pad** [2]
- char **state**
- char **node**
- short **age**
- char **reserved1**
- char **rssi**
- char **rate**
- char **reserved2**

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

- src/drivers/prism54.h

14.269 obj_stakey Struct Reference

Data Fields

- char **address** [6]
- char **keyid**
- char **reserved**
- short **options**
- char **type**
- char **length**
- char **key** [32]

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

- src/drivers/prism54.h

14.270 obj_stasc Struct Reference

Data Fields

- char **address** [6]
- char **keyid**
- char **tx_sc**
- unsigned long **sc_high**
- unsigned short **sc_low**

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

- src/drivers/prism54.h

14.271 oob_conf_data Struct Reference

Public Types

- enum { **OOB_METHOD_UNKNOWN** = 0, **OOB_METHOD_DEV_PWD_E**, **OOB_METHOD_DEV_PWD_R**, **OOB_METHOD_CRED** }

Data Fields

- enum oob_conf_data:: { ... } **oob_method**
- struct wpabuf * **dev_password**
- struct wpabuf * **pubkey_hash**

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

- src/wps/wps.h

14.272 oob_device_data Struct Reference

Data Fields

- char * **device_name**
- char * **device_path**
- void *(* **init_func**)(struct wps_context *, struct oob_device_data *, int)
- struct wpabuf *(* **read_func**)(void *)
- int(* **write_func**)(void *, struct wpabuf *)
- void(* **deinit_func**)(void *)

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

- src/wps/wps.h

14.273 oob_nfc_device_data Struct Reference

Data Fields

- int(∗ **init_func**)(char ∗)
- void ∗(∗ **read_func**)(size_t ∗)
- int(∗ **write_func**)(void ∗, size_t)
- void(∗ **deinit_func**)(void)

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

- src/wps/wps.h

14.274 os_time Struct Reference

Data Fields

- os_time_t **sec**
- os_time_t **usec**

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

- src/utils/os.h

14.275 pac_tlv_hdr Struct Reference

Data Fields

- be16 **type**
- be16 **len**

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

- src/eap_common/eap_fast_common.h

14.276 parse_data Struct Reference

Data Fields

- char * **name**
- int(* **parser**)(const struct parse_data *data, struct wpa_ssid *ssid, int line, const char *value)
- char *(* **writer**)(const struct parse_data *data, struct wpa_ssid *ssid)
- void * **param1**
- void * **param2**
- void * **param3**
- void * **param4**
- int **key_data**

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

- wpa_supplicant/config.c

14.277 pimdev_hdr_s Struct Reference

Data Fields

- int **op**
- unsigned long **oid**

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

- src/drivers/prism54.h

14.278 pkcs5_params Struct Reference

Public Types

- enum **pkcs5_alg** { **PKCS5_ALG_UNKNOWN**, **PKCS5_ALG_MD5_DES_CBC** }

Data Fields

- enum pkcs5_params::pkcs5_alg **alg**
- u8 **salt** [8]
- size_t **salt_len**
- unsigned int **iter_count**

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

- src/tls/pkcs5.c

14.279 PMKID_CANDIDATE Struct Reference

Data Fields

- NDIS_802_11_MAC_ADDRESS **BSSID**
- ULONG **Flags**

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

- src/drivers/driver_ndis.c

14.280 wpa_event_data::pmkid_candidate Struct Reference

Data for EVENT_PMKID_CANDIDATE.

`#include <driver.h>`

Data Fields

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

14.280.1 Detailed Description

Data for EVENT_PMKID_CANDIDATE.

14.280.2 Field Documentation

14.280.2.1 u8 wpa_event_data::pmkid_candidate::bssid[ETH_ALEN]

BSSID of the PMKID candidate

14.280.2.2 int wpa_event_data::pmkid_candidate::index

Smaller the index, higher the priority

14.280.2.3 int wpa_event_data::pmkid_candidate::preauth

Whether RSN IE includes pre-authenticate flag

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

- src/drivers/driver.h

14.281 preauth_test_data Struct Reference

Data Fields

- int **auth_timed_out**

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

- wpa_supplicant/preauth_test.c

14.282 prism2_download_param::prism2_download_area Struct Reference

Data Fields

- u32 **addr**
- u32 **len**
- caddr_t **ptr**

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

- src/drivers/driver_hostap.h

14.283 prism2_download_param Struct Reference

Data Structures

- struct prism2_download_area

Data Fields

- u32 **dl_cmd**
- u32 **start_addr**
- u32 **num_areas**
- struct prism2_download_param::prism2_download_area **data** [0]

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

- src/drivers/driver_hostap.h

14.284 prism2_hostapd_param Struct Reference

Data Fields

- u32 **cmd**
- u8 **sta_addr** [ETH_ALEN]
- union {
  struct {
  u16 **aid**
  u16 **capability**
  u8 **tx_supp_rates**
  } **add_sta**
  struct {
  u32 **inactive_sec**
  } **get_info_sta**
  struct {
  u8 **alg** [HOSTAP_CRYPT_ALG_NAME_LEN]
  u32 **flags**
  u32 **err**
  u8 **idx**
  u8 **seq** [8]
  u16 **key_len**
  u8 **key** [0]
  } **crypt**
  struct {
  u32 **flags_and**
  u32 **flags_or**
  } **set_flags_sta**
  struct {
  u16 **rid**
  u16 **len**
  u8 **data** [0]
  } **rid**
  struct {
  u8 **len**
  u8 **data** [0]
  } **generic_elem**
  struct {
  u16 **cmd**
  u16 **reason_code**
  } **mlme**
  struct {
  u8 **ssid_len**
  u8 **ssid** [32]
  } **scan_req**
  } **u**

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

- src/drivers/driver_hostap.h

14.285 privsep_cmd_associate Struct Reference

Data Fields

- u8 **bssid** [ETH_ALEN]
- u8 **ssid** [32]
- size_t **ssid_len**
- int **freq**
- int **pairwise_suite**
- int **group_suite**
- int **key_mgmt_suite**
- int **auth_alg**
- int **mode**
- size_t **wpa_ie_len**

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

- src/common/privsep_commands.h

14.286 privsep_cmd_set_key Struct Reference

Data Fields

- int **alg**
- u8 **addr** [ETH_ALEN]
- int **key_idx**
- int **set_tx**
- u8 **seq** [8]
- size_t **seq_len**
- u8 **key** [32]
- size_t **key_len**

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

- src/common/privsep_commands.h

14.287 prune_data Struct Reference

Data Fields

- struct hostapd_data * **hapd**
- const u8 * **addr**

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

- hostapd/drv_callbacks.c

14.288 radius_attr_data Struct Reference

Data Fields

- u8 * **data**
- size_t **len**

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

- src/radius/radius.h

14.289 radius_attr_hdr Struct Reference

Data Fields

- u8 **type**
- u8 **length**

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

- src/radius/radius.h

14.290 radius_attr_type Struct Reference

Public Types

- enum {

  RADIUS_ATTR_UNDIST, **RADIUS_ATTR_TEXT**, **RADIUS_ATTR_IP**, **RADIUS_ATTR_-HEXDUMP**,

  RADIUS_ATTR_INT32, **RADIUS_ATTR_IPV6** }

Data Fields

- u8 **type**
- char * **name**
- enum radius_attr_type:: { ... } **data_type**

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

- src/radius/radius.c

14.291 radius_attr_vendor Struct Reference

Data Fields

- u8 **vendor_type**
- u8 **vendor_length**

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

- src/radius/radius.h

14.292 radius_class_data Struct Reference

Data Fields

- struct radius_attr_data * **attr**
- size_t **count**

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

- src/radius/radius.h

14.293 radius_client Struct Reference

Data Fields

- struct radius_client * **next**
- struct in_addr **addr**
- struct in_addr **mask**
- char * **shared_secret**
- int **shared_secret_len**
- struct radius_session * **sessions**
- struct radius_server_counters **counters**

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

- src/radius/radius_server.c

14.294 radius_client_data Struct Reference

Internal RADIUS client data.

Data Fields

- void * ctx

  Context pointer for hostapd_logger() callbacks.

- struct hostapd_radius_servers * conf

  RADIUS client configuration (list of RADIUS servers to use).

- int auth_serv_sock

  Socket for authentication RADIUS messages.

- int acct_serv_sock

  Socket for accounting RADIUS messages.

- int **auth_serv_sock6**
- int **acct_serv_sock6**
- int auth_sock

  Current used socket for RADIUS authentication server.

- int acct_sock

  Current used socket for RADIUS accounting server.

- struct radius_rx_handler * **auth_handlers**
- size_t **num_auth_handlers**
- struct radius_rx_handler * **acct_handlers**
- size_t **num_acct_handlers**
- struct radius_msg_list * **msgs**
- size_t **num_msgs**
- u8 **next_radius_identifier**

14.294.1 Detailed Description

Internal RADIUS client data. This data structure is used internally inside the RADIUS client module. External users allocate this by calling radius_client_init() and free it by calling radius_client_deinit(). The pointer to this opaque data is used in calls to other functions as an identifier for the RADIUS client instance.

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

- src/radius/radius_client.c

14.295 radius_hdr Struct Reference

Data Fields

- u8 **code**
- u8 **identifier**
- u16 **length**
- u8 **authenticator** [16]

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

- src/radius/radius.h

14.296 radius_ms_mppe_keys Struct Reference

Data Fields

- u8 * **send**
- size_t **send_len**
- u8 * **recv**
- size_t **recv_len**

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

- src/radius/radius.h

14.297 radius_msg Struct Reference

Data Fields

- unsigned char * **buf**
- size_t **buf_size**
- size_t **buf_used**
- struct radius_hdr * **hdr**
- size_t * **attr_pos**
- size_t **attr_size**
- size_t **attr_used**

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

- src/radius/radius.h

14.298 radius_msg_list Struct Reference

RADIUS client message retransmit list.

Data Fields

- u8 **addr** [ETH_ALEN]
- struct radius_msg * **msg**
- RadiusType **msg_type**
- os_time_t **first_try**
- os_time_t **next_try**
- int **attempts**
- int **next_wait**
- struct os_time **last_attempt**
- u8 * **shared_secret**
- size_t **shared_secret_len**
- struct radius_msg_list * **next**

14.298.1 Detailed Description

RADIUS client message retransmit list. This data structure is used internally inside the RADIUS client module to store pending RADIUS requests that may still need to be retransmitted.

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

- src/radius/radius_client.c

14.299 radius_rx_handler Struct Reference

RADIUS client RX handler.

Data Fields

- RadiusRxResult(∗ **handler**)(struct radius_msg ∗msg, struct radius_msg ∗req, const u8 ∗shared_secret, size_t shared_secret_len, void ∗data)
- void ∗ **data**

14.299.1 Detailed Description

RADIUS client RX handler. This data structure is used internally inside the RADIUS client module to store registered RX handlers. These handlers are registered by calls to radius_client_register() and unregistered when the RADIUS client is deinitizlized with a call to radius_client_deinit().

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

- src/radius/radius_client.c

14.300 radius_server_conf Struct Reference

Data Fields

- int **auth_port**
- char * **client_file**
- void * **conf_ctx**
- void * **eap_sim_db_priv**
- void * **ssl_ctx**
- u8 * **pac_opaque_encr_key**
- u8 * **eap_fast_a_id**
- size_t **eap_fast_a_id_len**
- char * **eap_fast_a_id_info**
- int **eap_fast_prov**
- int **pac_key_lifetime**
- int **pac_key_refresh_time**
- int **eap_sim_aka_result_ind**
- int **tnc**
- struct wps_context * **wps**
- int **ipv6**
- int(* **get_eap_user**)(void *ctx, const u8 *identity, size_t identity_len, int phase2, struct eap_user *user)
- const char * **eap_req_id_text**
- size_t **eap_req_id_text_len**

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

- src/radius/radius_server.h

14.301 radius_server_counters Struct Reference

Data Fields

- u32 **access_requests**
- u32 **invalid_requests**
- u32 **dup_access_requests**
- u32 **access_accepts**
- u32 **access_rejects**
- u32 **access_challenges**
- u32 **malformed_access_requests**
- u32 **bad_authenticators**
- u32 **packets_dropped**
- u32 **unknown_types**

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

- src/radius/radius_server.c

14.302 radius_server_data Struct Reference

Data Fields

- int **auth_sock**
- struct radius_client * **clients**
- unsigned int **next_sess_id**
- void * **conf_ctx**
- int **num_sess**
- void * **eap_sim_db_priv**
- void * **ssl_ctx**
- u8 * **pac_opaque_encr_key**
- u8 * **eap_fast_a_id**
- size_t **eap_fast_a_id_len**
- char * **eap_fast_a_id_info**
- int **eap_fast_prov**
- int **pac_key_lifetime**
- int **pac_key_refresh_time**
- int **eap_sim_aka_result_ind**
- int **tnc**
- struct wps_context * **wps**
- int **ipv6**
- struct os_time **start_time**
- struct radius_server_counters **counters**
- int(* **get_eap_user**)(void *ctx, const u8 *identity, size_t identity_len, int phase2, struct eap_user *user)
- char * **eap_req_id_text**
- size_t **eap_req_id_text_len**

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

- src/radius/radius_server.c

14.303 radius_session Struct Reference

Data Fields

- struct radius_session * **next**
- struct radius_client * **client**
- struct radius_server_data * **server**
- unsigned int **sess_id**
- struct eap_sm * **eap**
- struct eap_eapol_interface * **eap_if**
- struct radius_msg * **last_msg**
- char * **last_from_addr**
- int **last_from_port**
- struct sockaddr_storage **last_from**
- socklen_t **last_fromlen**
- u8 **last_identifier**
- struct radius_msg * **last_reply**
- u8 **last_authenticator** [16]

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

- src/radius/radius_server.c

14.304 radius_tunnel_attrs Struct Reference

Data Fields

- int **tag_used**
- int **type**
- int **medium_type**
- int **vlanid**

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

- src/radius/radius.c

14.305 recommended_tx_channel_width_action Struct Reference

Data Fields

- u8 **category**
- u8 **action**
- u8 **channel_width**

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

- src/common/ieee802_11_defs.h

14.306 rsn_error_kde Struct Reference

Data Fields

- be16 **mui**
- be16 **error_type**

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

- src/common/wpa_common.h

14.307 rsn_ie_hdr Struct Reference

Data Fields

- u8 **elem_id**
- u8 **len**
- u8 **version** [2]

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

- src/common/wpa_common.h

14.308 rsn_pmksa_cache Struct Reference

Data Fields

- struct rsn_pmksa_cache_entry * **pmkid** [PMKID_HASH_SIZE]
- struct rsn_pmksa_cache_entry * **pmksa**
- int **pmksa_count**
- void(* **free_cb**)(struct rsn_pmksa_cache_entry *entry, void *ctx)
- void * **ctx**
- struct wpa_sm * **sm**

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

- hostapd/pmksa_cache.c
- src/rsn_supp/pmksa_cache.c

14.309 rsn_pmksa_cache_entry Struct Reference

PMKSA cache entry.

`#include <pmksa_cache.h>`

Data Fields

- struct rsn_pmksa_cache_entry * **next**
- struct rsn_pmksa_cache_entry * **hnext**
- u8 **pmkid** [PMKID_LEN]
- u8 **pmk** [PMK_LEN]
- size_t **pmk_len**
- os_time_t **expiration**
- int **akmp**
- u8 **spa** [ETH_ALEN]
- u8 * **identity**
- size_t **identity_len**
- struct radius_class_data **radius_class**
- u8 **eap_type_authsrv**
- int **vlan_id**
- int **opportunistic**
- u8 **aa** [ETH_ALEN]
- os_time_t **reauth_time**
- void * network_ctx

  Network configuration context.

14.309.1 Detailed Description

PMKSA cache entry.

14.309.2 Field Documentation

14.309.2.1 void* rsn_pmksa_cache_entry::network_ctx

Network configuration context. This field is only used to match PMKSA cache entries to a specific network configuration (e.g., a specific SSID and security policy). This can be a pointer to the configuration entry, but PMKSA caching code does not dereference the value and this could be any kind of identifier.

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

- hostapd/pmksa_cache.h
- src/rsn_supp/pmksa_cache.h

14.310 rsn_pmksa_candidate Struct Reference

Data Fields

- struct rsn_pmksa_candidate * **next**
- u8 **bssid** [ETH_ALEN]
- int **priority**

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

- src/rsn_supp/preauth.c

14.311 rsn_supp_config Struct Reference

Data Fields

- void * **network_ctx**
- int **peerkey_enabled**
- int **allowed_pairwise_cipher**
- int **proactive_key_caching**
- int **eap_workaround**
- void * **eap_conf_ctx**
- const u8 * **ssid**
- size_t **ssid_len**
- int **wpa_ptk_rekey**

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

- src/rsn_supp/wpa.h

14.312 rtattr Struct Reference

Data Fields

- unsigned short **rta_len**
- unsigned short **rta_type**

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

- src/drivers/priv_netlink.h

14.313 scard_data Struct Reference

Data Fields

- SCARDCONTEXT **ctx**
- SCARDHANDLE **card**
- DWORD **protocol**
- sim_types **sim_type**
- int **pin1_required**

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

- src/utils/pcsc_funcs.c

14.314 secondary_channel_offset_ie Struct Reference

Data Fields

- u8 **id**
- u8 **length**
- u8 **secondary_offset_offset**

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

- src/common/ieee802_11_defs.h

14.315 security_parameters_st Struct Reference

Data Fields

- gnutls_connection_end_t **entity**
- gnutls_kx_algorithm_t **kx_algorithm**
- gnutls_cipher_algorithm_t **read_bulk_cipher_algorithm**
- gnutls_mac_algorithm_t **read_mac_algorithm**
- gnutls_compression_method_t **read_compression_algorithm**
- gnutls_cipher_algorithm_t **write_bulk_cipher_algorithm**
- gnutls_mac_algorithm_t **write_mac_algorithm**
- gnutls_compression_method_t **write_compression_algorithm**
- cipher_suite_st **current_cipher_suite**
- opaque **master_secret** [TLS_MASTER_SIZE]
- opaque **client_random** [TLS_RANDOM_SIZE]
- opaque **server_random** [TLS_RANDOM_SIZE]

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

- src/crypto/tls_gnutls.c

14.316 SHA1Context Struct Reference

Data Fields

- u32 **state** [5]
- u32 **count** [2]
- unsigned char **buffer** [64]

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

- src/crypto/sha1_i.h

14.317 sha256_state Struct Reference

Data Fields

- u64 **length**
- u32 **state** [8]
- u32 **curlen**
- u8 **buf** [64]

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

- src/crypto/sha256-internal.c

14.318 sockaddr_nl Struct Reference

Data Fields

- sa_family_t **nl_family**
- unsigned short **nl_pad**
- u32 **nl_pid**
- u32 **nl_groups**

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

- src/drivers/priv_netlink.h

14.319 sta_id_search Struct Reference

Data Fields

- u8 **identifier**
- struct eapol_state_machine * **sm**

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

- hostapd/ieee802_1x.c

14.320 sta_info Struct Reference

Public Types

- enum { **STA_NULLFUNC** = 0, **STA_DISASSOC**, **STA_DEAUTH**, **STA_REMOVE** }

Data Fields

- struct sta_info * **next**
- struct sta_info * **hnext**
- u8 **addr** [6]
- u16 **aid**
- u32 **flags**
- u16 **capability**
- u16 **listen_interval**
- u8 **supported_rates** [WLAN_SUPP_RATES_MAX]
- int **supported_rates_len**
- unsigned int **nonerp_set**:1
- unsigned int **no_short_slot_time_set**:1
- unsigned int **no_short_preamble_set**:1
- unsigned int **no_ht_gf_set**:1
- unsigned int **no_ht_set**:1
- unsigned int **ht_20mhz_set**:1
- u16 **auth_alg**
- u8 **previous_ap** [6]
- enum sta_info:: { ... } **timeout_next**
- struct eapol_state_machine * **eapol_sm**
- struct ieee80211_mgmt * **last_assoc_req**
- u32 **acct_session_id_hi**
- u32 **acct_session_id_lo**
- time_t **acct_session_start**
- int **acct_session_started**
- int **acct_terminate_cause**
- int **acct_interim_interval**
- unsigned long **last_rx_bytes**
- unsigned long **last_tx_bytes**
- u32 **acct_input_gigawords**
- u32 **acct_output_gigawords**
- u8 * **challenge**
- struct wpa_state_machine * **wpa_sm**
- struct rsn_preauth_interface * **preauth_iface**
- struct hostapd_ssid * **ssid**
- struct hostapd_ssid * **ssid_probe**
- int **vlan_id**
- struct wpabuf * **wps_ie**

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

- hostapd/sta_info.h

14.321 wpa_event_data::stkstart Struct Reference

Data for EVENT_STKSTART.

`#include <driver.h>`

Data Fields

- u8 **peer** [ETH_ALEN]

14.321.1 Detailed Description

Data for EVENT_STKSTART.

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

- src/drivers/driver.h

14.322 subscr_addr Struct Reference

Data Fields

- struct subscr_addr * **next**
- struct subscr_addr * **prev**
- struct subscription * **s**
- char * **domain_and_port**
- char * **path**
- struct sockaddr_in **saddr**

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

- src/wps/wps_upnp_i.h

14.323 subscription Struct Reference

Data Fields

- struct subscription * **next**
- struct subscription * **prev**
- struct upnp_wps_device_sm * **sm**
- time_t **timeout_time**
- unsigned **next_subscriber_sequence**
- u8 **uuid** [UUID_LEN]
- struct subscr_addr * **addr_list**
- int **n_addr**
- struct wps_event_ * **event_queue**
- int **n_queue**
- struct wps_event_ * **current_event**

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

- src/wps/wps_upnp_i.h

14.324 test_client_socket Struct Reference

Data Fields

- struct test_client_socket * **next**
- u8 **addr** [ETH_ALEN]
- struct sockaddr_un **un**
- socklen_t **unlen**
- struct test_driver_bss * **bss**

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

- src/drivers/driver_test.c

14.325 test_driver_bss Struct Reference

Data Fields

- struct test_driver_bss * **next**
- char **ifname** [IFNAMSIZ+1]
- u8 **bssid** [ETH_ALEN]
- u8 * **ie**
- size_t **ielen**
- u8 * **wps_beacon_ie**
- size_t **wps_beacon_ie_len**
- u8 * **wps_probe_resp_ie**
- size_t **wps_probe_resp_ie_len**
- u8 **ssid** [32]
- size_t **ssid_len**
- int **privacy**

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

- src/drivers/driver_test.c

14.326 wpa_event_data::timeout_event Struct Reference

Data Fields

- u8 **addr** [ETH_ALEN]

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

- src/drivers/driver.h

14.327 tls_cipher_data Struct Reference

Data Fields

- tls_cipher **cipher**
- tls_cipher_type **type**
- size_t **key_material**
- size_t **expanded_key_material**
- size_t **block_size**
- enum crypto_cipher_alg **alg**

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

- src/tls/tlsv1_common.h

14.328 tls_cipher_suite Struct Reference

Data Fields

- u16 **suite**
- tls_key_exchange **key_exchange**
- tls_cipher **cipher**
- tls_hash **hash**

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

- src/tls/tlsv1_common.h

14.329 tls_config Struct Reference

Data Fields

- const char * **opensc_engine_path**
- const char * **pkcs11_engine_path**
- const char * **pkcs11_module_path**
- int **fips_mode**

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

- src/crypto/tls.h

14.330 tls_connection Struct Reference

Data Fields

- gnutls_session **session**
- char * **subject_match**
- char * **altsubject_match**
- int **read_alerts**
- int **write_alerts**
- int **failed**
- u8 * **pre_shared_secret**
- size_t **pre_shared_secret_len**
- int **established**
- int **verify_peer**
- u8 * **push_buf**
- u8 * **pull_buf**
- u8 * **pull_buf_offset**
- size_t **push_buf_len**
- size_t **pull_buf_len**
- int **params_set**
- gnutls_certificate_credentials_t **xcred**
- int **tls_ia**
- int **final_phase_finished**
- struct tlsv1_client * **client**
- struct tlsv1_server * **server**
- PRFileDesc * **fd**
- SSL * **ssl**
- BIO * **ssl_in**
- BIO * **ssl_out**
- ENGINE * **engine**
- EVP_PKEY * **private_key**
- tls_session_ticket_cb **session_ticket_cb**
- void * **session_ticket_cb_ctx**
- u8 * **session_ticket**
- size_t **session_ticket_len**
- int **start**
- SCHANNEL_CRED **schannel_cred**
- CredHandle **creds**
- CtxtHandle **context**
- u8 **eap_tls_prf** [128]
- int **eap_tls_prf_set**

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

- src/crypto/tls_gnutls.c
- src/crypto/tls_internal.c
- src/crypto/tls_nss.c
- src/crypto/tls_openssl.c
- src/crypto/tls_schannel.c

14.331 tls_connection_params Struct Reference

Parameters for TLS connection.

`#include <tls.h>`

Data Fields

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

14.331.1 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 id when using engine (this is OpenSSL specific for now)

cert_id the certificate's id when using engine

ca_cert_id the CA certificate's id when using engine

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

TLS connection parameters to be configured with tls_connection_set_params() and tls_global_set_params().

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

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

- src/crypto/tls.h

14.332 tls_global Struct Reference

Data Fields

- void * **session_data**
- size_t **session_data_size**
- int **server**
- int **params_set**
- gnutls_certificate_credentials_t **xcred**
- struct tlsv1_credentials * **server_cred**
- int **check_crl**
- HMODULE **hsecurity**
- PSecurityFunctionTable **sspi**
- HCERTSTORE **my_cert_store**

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

- src/crypto/tls_gnutls.c
- src/crypto/tls_internal.c
- src/crypto/tls_schannel.c

14.333 tls_keys Struct Reference

Data Fields

- const u8 * **master_key**
- size_t **master_key_len**
- const u8 * **client_random**
- size_t **client_random_len**
- const u8 * **server_random**
- size_t **server_random_len**
- const u8 * **inner_secret**
- size_t **inner_secret_len**

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

- src/crypto/tls.h

14.334 tls_verify_hash Struct Reference

Data Fields

- struct crypto_hash * **md5_client**
- struct crypto_hash * **sha1_client**
- struct crypto_hash * **md5_server**
- struct crypto_hash * **sha1_server**
- struct crypto_hash * **md5_cert**
- struct crypto_hash * **sha1_cert**

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

- src/tls/tlsv1_common.h

14.335 tlsv1_client Struct Reference

Public Types

- enum {

  CLIENT_HELLO, **SERVER_HELLO**, **SERVER_CERTIFICATE**, **SERVER_KEY_EXCHANGE**,

  SERVER_CERTIFICATE_REQUEST, **SERVER_HELLO_DONE**, **CLIENT_KEY_EXCHANGE**, **CHANGE_CIPHER_SPEC**,

  SERVER_CHANGE_CIPHER_SPEC, **SERVER_FINISHED**, **ACK_FINISHED**, **ESTABLISHED**,

  FAILED }

Data Fields

- enum tlsv1_client:: { ... } **state**
- struct tlsv1_record_layer **rl**
- u8 **session_id** [TLS_SESSION_ID_MAX_LEN]
- size_t **session_id_len**
- u8 **client_random** [TLS_RANDOM_LEN]
- u8 **server_random** [TLS_RANDOM_LEN]
- u8 **master_secret** [TLS_MASTER_SECRET_LEN]
- u8 **alert_level**
- u8 **alert_description**
- unsigned int **certificate_requested**:1
- unsigned int **session_resumed**:1
- unsigned int **session_ticket_included**:1
- unsigned int **use_session_ticket**:1
- struct crypto_public_key * **server_rsa_key**
- struct tls_verify_hash **verify**
- u16 **cipher_suites** [MAX_CIPHER_COUNT]
- size_t **num_cipher_suites**
- u16 **prev_cipher_suite**
- u8 * **client_hello_ext**
- size_t **client_hello_ext_len**
- u8 * **dh_p**
- size_t **dh_p_len**
- u8 * **dh_g**
- size_t **dh_g_len**
- u8 * **dh_ys**
- size_t **dh_ys_len**
- struct tlsv1_credentials * **cred**
- tlsv1_client_session_ticket_cb **session_ticket_cb**
- void * **session_ticket_cb_ctx**

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

- src/tls/tlsv1_client_i.h

14.336 tlsv1_credentials Struct Reference

Data Fields

- struct x509_certificate * **trusted_certs**
- struct x509_certificate * **cert**
- struct crypto_private_key * **key**
- u8 * **dh_p**
- size_t **dh_p_len**
- u8 * **dh_g**
- size_t **dh_g_len**

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

- src/tls/tlsv1_cred.h

14.337 tlsv1_record_layer Struct Reference

Data Fields

- u8 **write_mac_secret** [TLS_MAX_WRITE_MAC_SECRET_LEN]
- u8 **read_mac_secret** [TLS_MAX_WRITE_MAC_SECRET_LEN]
- u8 **write_key** [TLS_MAX_WRITE_KEY_LEN]
- u8 **read_key** [TLS_MAX_WRITE_KEY_LEN]
- u8 **write_iv** [TLS_MAX_IV_LEN]
- u8 **read_iv** [TLS_MAX_IV_LEN]
- size_t **hash_size**
- size_t **key_material_len**
- size_t **iv_size**
- enum crypto_hash_alg **hash_alg**
- enum crypto_cipher_alg **cipher_alg**
- u8 **write_seq_num** [TLS_SEQ_NUM_LEN]
- u8 **read_seq_num** [TLS_SEQ_NUM_LEN]
- u16 **cipher_suite**
- u16 **write_cipher_suite**
- u16 **read_cipher_suite**
- struct crypto_cipher * **write_cbc**
- struct crypto_cipher * **read_cbc**

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

- src/tls/tlsv1_record.h

14.338 tlsv1_server Struct Reference

Public Types

- enum {

  CLIENT_HELLO, **SERVER_HELLO**, **SERVER_CERTIFICATE**, **SERVER_KEY_EXCHANGE**,

  SERVER_CERTIFICATE_REQUEST, **SERVER_HELLO_DONE**, **CLIENT_CERTIFICATE**, **CLIENT_KEY_EXCHANGE**,

  CERTIFICATE_VERIFY, **CHANGE_CIPHER_SPEC**, **CLIENT_FINISHED**, **SERVER_CHANGE_CIPHER_SPEC**,

  SERVER_FINISHED, **ESTABLISHED**, **FAILED** }

Data Fields

- enum tlsv1_server:: { ... } **state**
- struct tlsv1_record_layer **rl**
- u8 **session_id** [TLS_SESSION_ID_MAX_LEN]
- size_t **session_id_len**
- u8 **client_random** [TLS_RANDOM_LEN]
- u8 **server_random** [TLS_RANDOM_LEN]
- u8 **master_secret** [TLS_MASTER_SECRET_LEN]
- u8 **alert_level**
- u8 **alert_description**
- struct crypto_public_key * **client_rsa_key**
- struct tls_verify_hash **verify**
- u16 **cipher_suites** [MAX_CIPHER_COUNT]
- size_t **num_cipher_suites**
- u16 **cipher_suite**
- struct tlsv1_credentials * **cred**
- int **verify_peer**
- u16 **client_version**
- u8 * **session_ticket**
- size_t **session_ticket_len**
- tlsv1_server_session_ticket_cb **session_ticket_cb**
- void * **session_ticket_cb_ctx**
- int **use_session_ticket**
- u8 * **dh_secret**
- size_t **dh_secret_len**

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

- src/tls/tlsv1_server_i.h

14.339 tnc_if_imc Struct Reference

Data Fields

- struct tnc_if_imc * **next**
- char * **name**
- char * **path**
- void * **dlhandle**
- TNC_IMCID **imcID**
- TNC_ConnectionID **connectionID**
- TNC_MessageTypeList **supported_types**
- size_t **num_supported_types**
- u8 * **imc_send**
- size_t **imc_send_len**
- TNC_Result(* **Initialize**)(TNC_IMCID imcID, TNC_Version minVersion, TNC_Version maxVersion, TNC_Version *pOutActualVersion)
- TNC_Result(* **NotifyConnectionChange**)(TNC_IMCID imcID, TNC_ConnectionID connectionID, TNC_ConnectionState newState)
- TNC_Result(* **BeginHandshake**)(TNC_IMCID imcID, TNC_ConnectionID connectionID)
- TNC_Result(* **ReceiveMessage**)(TNC_IMCID imcID, TNC_ConnectionID connectionID, TNC_BufferReference messageBuffer, TNC_UInt32 messageLength, TNC_MessageType messageType)
- TNC_Result(* **BatchEnding**)(TNC_IMCID imcID, TNC_ConnectionID connectionID)
- TNC_Result(* **Terminate**)(TNC_IMCID imcID)
- TNC_Result(* **ProvideBindFunction**)(TNC_IMCID imcID, TNC_TNCC_BindFunctionPointer bindFunction)

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

- src/eap_peer/tncc.c

14.340 tnc_if_imv Struct Reference

Data Fields

- struct tnc_if_imv * **next**
- char * **name**
- char * **path**
- void * **dlhandle**
- TNC_IMVID **imvID**
- TNC_MessageTypeList **supported_types**
- size_t **num_supported_types**
- TNC_Result(* **Initialize**)(TNC_IMVID imvID, TNC_Version minVersion, TNC_Version maxVersion, TNC_Version *pOutActualVersion)
- TNC_Result(* **NotifyConnectionChange**)(TNC_IMVID imvID, TNC_ConnectionID connectionID, TNC_ConnectionState newState)
- TNC_Result(* **ReceiveMessage**)(TNC_IMVID imvID, TNC_ConnectionID connectionID, TNC_BufferReference message, TNC_UInt32 messageLength, TNC_MessageType messageType)
- TNC_Result(* **SolicitRecommendation**)(TNC_IMVID imvID, TNC_ConnectionID connectionID)
- TNC_Result(* **BatchEnding**)(TNC_IMVID imvID, TNC_ConnectionID connectionID)
- TNC_Result(* **Terminate**)(TNC_IMVID imvID)
- TNC_Result(* **ProvideBindFunction**)(TNC_IMVID imvID, TNC_TNCS_BindFunctionPointer bindFunction)

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

- src/eap_server/tncs.c

14.341 tncc_data Struct Reference

Data Fields

- struct tnc_if_imc * **imc**
- unsigned int **last_batchid**

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

- src/eap_peer/tncc.c

14.342 tncs_data Struct Reference

Data Structures

- struct conn_imv

Data Fields

- struct tncs_data * **next**
- struct tnc_if_imv * **imv**
- TNC_ConnectionID **connectionID**
- unsigned int **last_batchid**
- enum IMV_Action_Recommendation **recommendation**
- int **done**
- struct tncs_data::conn_imv **imv_data** [TNC_MAX_IMV_ID]
- char * **tncs_message**

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

- src/eap_server/tncs.c

14.343 tncs_global Struct Reference

Data Fields

- struct tnc_if_imv * **imv**
- TNC_ConnectionID **next_conn_id**
- struct tncs_data * **connections**

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

- src/eap_server/tncs.c

14.344 ttls_avp Struct Reference

Data Fields

- be32 **avp_code**
- be32 **avp_length**

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

- src/eap_common/eap_ttls.h

14.345 ttls_avp_vendor Struct Reference

Data Fields

- be32 **avp_code**
- be32 **avp_length**
- be32 **vendor_id**

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

- src/eap_common/eap_ttls.h

14.346 ttls_parse_avp Struct Reference

Data Fields

- u8 * **mschapv2**
- u8 * **eapdata**
- size_t **eap_len**
- int **mschapv2_error**

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

- src/eap_peer/eap_ttls.c

14.347 upnp_pending_message Struct Reference

Pending PutWLANResponse messages.

`#include <wps.h>`

Data Fields

- struct upnp_pending_message * **next**
- u8 **addr** [ETH_ALEN]
- struct wpabuf * **msg**
- enum wps_msg_type **type**

14.347.1 Detailed Description

Pending PutWLANResponse messages.

Parameters:

next Pointer to next pending message or NULL

addr NewWLANEventMAC

msg NewMessage

type Message Type

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

- src/wps/wps.h

14.348 upnp_wps_device_ctx Struct Reference

Data Fields

- struct wpabuf *(* **rx_req_get_device_info**)(void *priv, struct upnp_wps_peer *peer)
- struct wpabuf *(* **rx_req_put_message**)(void *priv, struct upnp_wps_peer *peer, const struct wpabuf *msg)
- int(* **rx_req_put_wlan_response**)(void *priv, enum upnp_wps_wlanevent_type ev_type, const u8 *mac_addr, const struct wpabuf *msg, enum wps_msg_type msg_type)
- int(* **rx_req_set_selected_registrar**)(void *priv, const struct wpabuf *msg)

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

- src/wps/wps_upnp.h

14.349 upnp_wps_device_sm Struct Reference

Data Fields

- struct upnp_wps_device_ctx * **ctx**
- struct wps_context * **wps**
- void * **priv**
- char * **root_dir**
- char * **desc_url**
- int **started**
- char * **net_if**
- char * **mac_addr_text**
- u8 **mac_addr** [ETH_ALEN]
- char * **ip_addr_text**
- unsigned **ip_addr**
- int **multicast_sd**
- int **ssdp_sd**
- int **ssdp_sd_registered**
- unsigned **advertise_count**
- struct advertisement_state_machine **advertisement**
- struct advertisement_state_machine * **msearch_replies**
- int **n_msearch_replies**
- int **web_port**
- struct http_server * **web_srv**
- struct subscription * **subscriptions**
- int **n_subscriptions**
- int **event_send_all_queued**
- char * **wlanevent**
- struct upnp_wps_peer **peer**

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

- src/wps/wps_upnp_i.h

14.350 upnp_wps_peer Struct Reference

Data Fields

- struct wps_data * **wps**

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

- src/wps/wps_upnp.h

14.351 wext_scan_data Struct Reference

Data Fields

- struct wpa_scan_res **res**
- u8 * **ie**
- size_t **ie_len**
- u8 **ssid** [32]
- size_t **ssid_len**
- int **maxrate**

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

- src/drivers/driver_wext.c

14.352 wiphy_info_data Struct Reference

Data Fields

- int **max_scan_ssids**
- int **ap_supported**
- int **auth_supported**
- int **connect_supported**

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

- src/drivers/driver_nl80211.c

14.353 WirelessInfo Struct Reference

Data Fields

- UInt16 **link_qual**
- UInt16 **comms_qual**
- UInt16 **signal**
- UInt16 **noise**
- UInt16 **port_stat**
- UInt16 **client_mode**
- UInt16 **res1**
- UInt16 **power**
- UInt16 **res2**
- UInt8 **bssID** [6]
- UInt8 **ssid** [34]

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

- src/drivers/Apple80211.h

14.354 WirelessInfo2 Struct Reference

Data Fields

- WirelessInfo **info1**
- UInt8 **macAddress** [6]

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

- src/drivers/Apple80211.h

14.355 WirelessNetworkInfo Struct Reference

Data Fields

- UInt16 **channel**
- UInt16 **noise**
- UInt16 **signal**
- UInt8 **bssid** [6]
- UInt16 **beacon_int**
- UInt16 **capability**
- UInt16 **ssid_len**
- UInt8 **ssid** [32]

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

- src/drivers/Apple80211.h

14.356 wlc_deauth_t Struct Reference

Data Fields

- uint32 **val**
- struct ether_addr **ea**
- uint16 **res**

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

- src/drivers/driver_broadcom.c

14.357 wmm_ac_parameter Struct Reference

Data Fields

- u8 **aci_aifsn**
- u8 **cw**
- le16 **txop_limit**

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

- src/common/ieee802_11_defs.h

14.358 wmm_information_element Struct Reference

Data Fields

- u8 **oui** [3]
- u8 **oui_type**
- u8 **oui_subtype**
- u8 **version**
- u8 **qos_info**

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

- src/common/ieee802_11_defs.h

14.359 wmm_parameter_element Struct Reference

Data Fields

- u8 **oui** [3]
- u8 **oui_type**
- u8 **oui_subtype**
- u8 **version**
- u8 **qos_info**
- u8 **reserved**
- struct wmm_ac_parameter **ac** [4]

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

- src/common/ieee802_11_defs.h

14.360 wmm_tspec_element Struct Reference

Data Fields

- u8 **eid**
- u8 **length**
- u8 **oui** [3]
- u8 **oui_type**
- u8 **oui_subtype**
- u8 **version**
- u8 **ts_info** [3]
- le16 **nominal_msdu_size**
- le16 **maximum_msdu_size**
- le32 **minimum_service_interval**
- le32 **maximum_service_interval**
- le32 **inactivity_interval**
- le32 **suspension_interval**
- le32 **service_start_time**
- le32 **minimum_data_rate**
- le32 **mean_data_rate**
- le32 **peak_data_rate**
- le32 **maximum_burst_size**
- le32 **delay_bound**
- le32 **minimum_phy_rate**
- le16 **surplus_bandwidth_allowance**
- le16 **medium_time**

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

- src/common/ieee802_11_defs.h

14.361 wpa_assoc_info Struct Reference

Data Fields

- const u8 * **bssid**
- const u8 * **ssid**
- size_t **ssid_len**
- int **freq**
- const u8 * **wpa_ie**
- size_t **wpa_ie_len**
- wpa_cipher **pairwise_suite**
- wpa_cipher **group_suite**
- wpa_key_mgmt **key_mgmt_suite**
- int **auth_alg**
- int **mode**

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

- src/drivers/driver_ndiswrapper.c

14.362 wpa_auth_callbacks Struct Reference

Data Fields

- void ∗ **ctx**
- void(∗ **logger**)(void ∗ctx, const u8 ∗addr, logger_level level, const char ∗txt)
- void(∗ **disconnect**)(void ∗ctx, const u8 ∗addr, u16 reason)
- void(∗ **mic_failure_report**)(void ∗ctx, const u8 ∗addr)
- void(∗ **set_eapol**)(void ∗ctx, const u8 ∗addr, wpa_eapol_variable var, int value)
- int(∗ **get_eapol**)(void ∗ctx, const u8 ∗addr, wpa_eapol_variable var)
- const u8 ∗(∗ **get_psk**)(void ∗ctx, const u8 ∗addr, const u8 ∗prev_psk)
- int(∗ **get_msk**)(void ∗ctx, const u8 ∗addr, u8 ∗msk, size_t ∗len)
- int(∗ **set_key**)(void ∗ctx, int vlan_id, wpa_alg alg, const u8 ∗addr, int idx, u8 ∗key, size_t key_len)
- int(∗ **get_seqnum**)(void ∗ctx, const u8 ∗addr, int idx, u8 ∗seq)
- int(∗ **get_seqnum_igtk**)(void ∗ctx, const u8 ∗addr, int idx, u8 ∗seq)
- int(∗ **send_eapol**)(void ∗ctx, const u8 ∗addr, const u8 ∗data, size_t data_len, int encrypt)
- int(∗ **for_each_sta**)(void ∗ctx, int(∗cb)(struct wpa_state_machine ∗sm, void ∗ctx), void ∗cb_ctx)
- int(∗ **for_each_auth**)(void ∗ctx, int(∗cb)(struct wpa_authenticator ∗a, void ∗ctx), void ∗cb_ctx)
- int(∗ **send_ether**)(void ∗ctx, const u8 ∗dst, u16 proto, const u8 ∗data, size_t data_len)

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

- hostapd/wpa.h

14.363 wpa_auth_config Struct Reference

Data Fields

- int **wpa**
- int **wpa_key_mgmt**
- int **wpa_pairwise**
- int **wpa_group**
- int **wpa_group_rekey**
- int **wpa_strict_rekey**
- int **wpa_gmk_rekey**
- int **wpa_ptk_rekey**
- int **rsn_pairwise**
- int **rsn_preauth**
- int **eapol_version**
- int **peerkey**
- int **wmm_enabled**
- int **okc**

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

- hostapd/wpa.h

14.364 wpa_auth_iface_iter_data Struct Reference

Data Fields

- int(∗ **cb**)(struct wpa_authenticator ∗sm, void ∗ctx)
- void ∗ **cb_ctx**

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

- hostapd/hostapd.c

14.365 wpa_auth_okc_iter_data Struct Reference

Data Fields

- struct rsn_pmksa_cache_entry * **pmksa**
- const u8 * **aa**
- const u8 * **spa**
- const u8 * **pmkid**

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

- hostapd/wpa_auth_ie.c

14.366 wpa_authenticator Struct Reference

Data Fields

- struct wpa_group * **group**
- unsigned int **dot11RSNAStatsTKIPRemoteMICFailures**
- u32 **dot11RSNAAuthenticationSuiteSelected**
- u32 **dot11RSNAPairwiseCipherSelected**
- u32 **dot11RSNAGroupCipherSelected**
- u8 **dot11RSNAPMKIDUsed** [PMKID_LEN]
- u32 **dot11RSNAAuthenticationSuiteRequested**
- u32 **dot11RSNAPairwiseCipherRequested**
- u32 **dot11RSNAGroupCipherRequested**
- unsigned int **dot11RSNATKIPCounterMeasuresInvoked**
- unsigned int **dot11RSNA4WayHandshakeFailures**
- struct wpa_stsl_negotiation * **stsl_negotiations**
- struct wpa_auth_config **conf**
- struct wpa_auth_callbacks **cb**
- u8 * **wpa_ie**
- size_t **wpa_ie_len**
- u8 **addr** [ETH_ALEN]
- struct rsn_pmksa_cache * **pmksa**
- struct wpa_ft_pmk_cache * **ft_pmk_cache**

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

- hostapd/wpa_auth_i.h

14.367 wpa_blacklist Struct Reference

Data Fields

- struct wpa_blacklist * **next**
- u8 **bssid** [ETH_ALEN]
- int **count**

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

- wpa_supplicant/blacklist.h

14.368 wpa_cli_cmd Struct Reference

Data Fields

- const char * **cmd**
- int(* **handler**)(struct wpa_ctrl *ctrl, int argc, char *argv[ ])
- enum wpa_cli_cmd_flags **flags**
- const char * **usage**

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

- wpa_supplicant/wpa_cli.c

14.369 wpa_client_mlme Struct Reference

Data Fields

- int **dummy**

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

- wpa_supplicant/wpa_supplicant_i.h

14.370 wpa_config Struct Reference

wpa_supplicant configuration data

`#include <config.h>`

Data Fields

- struct wpa_ssid * ssid
  Head of the global network list.
- struct wpa_ssid ** pssid
  Per-priority network lists (in priority order).
- int num_prio
  Number of different priorities used in the pssid lists.
- int eapol_version
  IEEE 802.1X/EAPOL version number.
- int ap_scan
  AP scanning/selection.
- char * ctrl_interface
  Parameters for the control interface.
- char * ctrl_interface_group
  Control interface group (DEPRECATED).
- int fast_reauth
  EAP fast re-authentication (session resumption).
- char * driver_param
  Driver interface parameters.
- unsigned int dot11RSNAConfigPMKLifetime
  Maximum lifetime of a PMK.
- unsigned int dot11RSNAConfigPMKReauthThreshold
  PMK re-authentication threshold.
- unsigned int dot11RSNAConfigSATimeout
  Security association timeout.
- int update_config
  Is wpa_supplicant allowed to update configuration.
- struct wpa_config_blob * blobs
  Configuration blobs.

- u8 uuid [16]

  Universally Unique IDentifier (UUID; see RFC 4122) for WPS.

- char * device_name

  Device Name (WPS).

- char * manufacturer

  Manufacturer (WPS).

- char * model_name

  Model Name (WPS).

- char * model_number

  Model Number (WPS).

- char * serial_number

  Serial Number (WPS).

- char * device_type

  Primary Device Type (WPS).

- u8 os_version [4]

  OS Version (WPS).

- char country [2]

  Country code.

- int wps_cred_processing

  Credential processing.

14.370.1 Detailed Description

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

14.370.2 Field Documentation

14.370.2.1 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.

14.370.2.2 char wpa_config::country[2]

Country code. This is the ISO/IEC alpha2 country code for which we are operating in

14.370.2.3 char* wpa_config::ctrl_interface

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

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

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

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

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

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

14.370.2.4 char* wpa_config::ctrl_interface_group

Control interface group (DEPRECATED). This variable is only used for backwards compatibility. Group for UNIX domain sockets should now be specified using GROUP=group in ctrl_interface variable.

14.370.2.5 char* wpa_config::device_name

Device Name (WPS). User-friendly description of device; up to 32 octets encoded in UTF-8

14.370.2.6 char* wpa_config::device_type

Primary Device Type (WPS). Used format: categ-OUI-subcateg categ = Category as an integer value OUI = OUI and type octet as a 4-octet hex-encoded value; 0050F204 for default WPS OUI subcateg = OUI-specific Sub Category as an integer value Examples: 1-0050F204-1 (Computer / PC) 1-0050F204-2 (Computer / Server) 5-0050F204-1 (Storage / NAS) 6-0050F204-1 (Network Infrastructure / AP)

14.370.2.7 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).

14.370.2.8 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.

14.370.2.9 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).

14.370.2.10 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.

14.370.2.11 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).

14.370.2.12 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.

14.370.2.13 char* wpa_config::manufacturer

Manufacturer (WPS). The manufacturer of the device (up to 64 ASCII characters)

14.370.2.14 char* wpa_config::model_name

Model Name (WPS). Model of the device (up to 32 ASCII characters)

14.370.2.15 char* wpa_config::model_number

Model Number (WPS). Additional device description (up to 32 ASCII characters)

14.370.2.16 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.

14.370.2.17 u8 wpa_config::os_version[4]

OS Version (WPS). 4-octet operating system version number

14.370.2.18 char* wpa_config::serial_number

Serial Number (WPS). Serial number of the device (up to 32 characters)

14.370.2.19 struct wpa_ssid* wpa_config::ssid [read]

Head of the global network list. This is the head for the list of all the configured networks.

14.370.2.20 int wpa_config::update_config

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

14.370.2.21 int wpa_config::wps_cred_processing

Credential processing. 0 = process received credentials internally 1 = do not process received credentials; just pass them over ctrl_iface to external program(s) 2 = process received credentials internally and pass them over ctrl_iface to external program(s)

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

- wpa_supplicant/config.h

14.371 wpa_config_blob Struct Reference

Named configuration blob.

`#include <eap_config.h>`

Data Fields

- char * name

  Blob name.

- u8 * data

  Pointer to binary data.

- size_t len

  Length of binary data.

- struct wpa_config_blob * next

  Pointer to next blob in the configuration.

14.371.1 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.

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

- src/eap_peer/eap_config.h

14.372 wpa_ctrl Struct Reference

Internal structure for control interface library.

14.372.1 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.

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

- src/common/wpa_ctrl.c

14.373 wpa_ctrl_dst Struct Reference

Internal data structure of control interface clients.

Data Fields

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

14.373.1 Detailed Description

Internal data structure of control interface clients. 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_named_pipe.c and should not be touched directly from other files.

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

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

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

- hostapd/ctrl_iface.c
- wpa_supplicant/ctrl_iface_named_pipe.c
- wpa_supplicant/ctrl_iface_udp.c
- wpa_supplicant/ctrl_iface_unix.c

14.374 wpa_dbus_argument Struct Reference

Data Fields

- char * **name**
- char * **type**
- enum dbus_arg_direction **dir**

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

- wpa_supplicant/ctrl_iface_dbus_new_helpers.h

14.375 wpa_dbus_dict_entry Struct Reference

Data Fields

- int **type**
- int array_type
- const char * key
- union {
  char * **str_value**
  char **byte_value**
  dbus_bool_t **bool_value**
  dbus_int16_t **int16_value**
  dbus_uint16_t **uint16_value**
  dbus_int32_t **int32_value**
  dbus_uint32_t **uint32_value**
  dbus_int64_t **int64_value**
  dbus_uint64_t **uint64_value**
  double **double_value**
  char * **bytearray_value**
  char ** **strarray_value**
  };
- dbus_uint32_t **array_len**

14.375.1 Field Documentation

14.375.1.1 union { ... }

key of the dict entry Possible values of the property

14.375.1.2 int wpa_dbus_dict_entry::array_type

the dbus type of the dict entry's value

14.375.1.3 const char* wpa_dbus_dict_entry::key

the dbus type of the array elements if the dict entry value contains an array

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

- wpa_supplicant/dbus_dict_helpers.h

14.376 wpa_dbus_method_desc Struct Reference

DBus method description.

Data Fields

- struct wpa_dbus_method_desc * **next**
- char * **dbus_interface**
- char * **dbus_method**
- WPADBusMethodHandler **method_handler**
- void * **handler_argument**
- WPADBusArgumentFreeFunction **argument_free_func**
- int **args_num**
- struct wpa_dbus_argument **args** [ ]

14.376.1 Detailed Description

DBus method description.

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

- wpa_supplicant/ctrl_iface_dbus_new_helpers.c

14.377 wpa_dbus_object_desc Struct Reference

Data Fields

- DBusConnection * **connection**
- struct wpa_dbus_method_desc * **methods**
- struct wpa_dbus_signal_desc * **signals**
- struct wpa_dbus_property_desc * **properties**

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

- wpa_supplicant/ctrl_iface_dbus_new_helpers.h

14.378 wpa_dbus_property_desc Struct Reference

DBus property description.

Data Fields

- struct wpa_dbus_property_desc * **next**
- char * **dbus_interface**
- char * **dbus_property**
- char * **type**
- enum dbus_prop_access **access**
- WPADBusPropertyAccessor **getter**
- WPADBusPropertyAccessor **setter**
- void * **user_data**
- WPADBusArgumentFreeFunction **user_data_free_func**

14.378.1 Detailed Description

DBus property description.

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

- wpa_supplicant/ctrl_iface_dbus_new_helpers.c

14.379 wpa_dbus_signal_desc Struct Reference

DBus signal description.

Data Fields

- struct wpa_dbus_signal_desc * **next**
- char * **dbus_interface**
- char * **dbus_signal**
- int **args_num**
- struct wpa_dbus_argument **args** [0]

14.379.1 Detailed Description

DBus signal description.

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

- wpa_supplicant/ctrl_iface_dbus_new_helpers.c

14.380 wpa_driver_associate_params Struct Reference

Association parameters.

`#include <driver.h>`

Public Types

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

  IEEE 802.11w management frame protection.

Data Fields

- const u8 * bssid

  BSSID of the selected AP.

- const u8 * ssid

  The selected SSID.

- size_t **ssid_len**
- int freq

  Frequency of the channel the selected AP is using.

- const u8 * wpa_ie

  WPA information element for (Re)Association Request.

- size_t wpa_ie_len

  length of the wpa_ie

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

  Allowed authentication algorithms.

- int mode

  Operation mode (infra/ibss) IEEE80211_MODE_*.

- const u8 * wep_key [4]

  WEP keys for static WEP configuration.

- size_t wep_key_len [4]

  WEP key length for static WEP configuration.

- int wep_tx_keyidx

  WEP TX key index for static WEP configuration.

- enum wpa_driver_associate_params:: { ... } mgmt_frame_protection

IEEE 802.11w management frame protection.

- const u8 * ft_ies
  IEEE 802.11r / FT information elements.
- size_t ft_ies_len
  Length of ft_ies in bytes.
- const u8 * ft_md
  FT Mobility domain (6 octets) (also included inside ft_ies).
- const char * passphrase
  RSN passphrase for PSK.
- const u8 * psk
  RSN PSK (alternative for passphrase for PSK).
- int drop_unencrypted
  Enable/disable unencrypted frame filtering.
- const u8 * prev_bssid
  Previously used BSSID in this ESS.

14.380.1 Detailed Description

Association parameters. Data for struct wpa_driver_ops::associate().

14.380.2 Field Documentation

14.380.2.1 int wpa_driver_associate_params::auth_alg

Allowed authentication algorithms. Bit field of AUTH_ALG_*

14.380.2.2 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.

14.380.2.3 int wpa_driver_associate_params::drop_unencrypted

Enable/disable unencrypted frame filtering. 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.

14.380.2.4 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)

14.380.2.5 const u8* wpa_driver_associate_params::ft_ies

IEEE 802.11r / FT information elements. If the supplicant is using IEEE 802.11r (FT) and has the needed keys for fast transition, this parameter is set to include the IEs that are to be sent in the next FT Authentication Request message. update_ft_ies() handler is called to update the IEs for further FT messages in the sequence.

The driver should use these IEs only if the target AP is advertising the same mobility domain as the one included in the MDIE here.

In ap_scan=2 mode, the driver can use these IEs when moving to a new AP after the initial association. These IEs can only be used if the target AP is advertising support for FT and is using the same MDIE and SSID as the current AP.

The driver is responsible for reporting the FT IEs received from the AP's response using wpa_supplicant_event() with EVENT_FT_RESPONSE type. update_ft_ies() handler will then be called with the FT IEs to include in the next frame in the authentication sequence.

14.380.2.6 const u8* wpa_driver_associate_params::ft_md

FT Mobility domain (6 octets) (also included inside ft_ies). This value is provided to allow the driver interface easier access to the current mobility domain. This value is set to NULL if no mobility domain is currently active.

14.380.2.7 const char* wpa_driver_associate_params::passphrase

RSN passphrase for PSK. This value is made available only for WPA/WPA2-Personal (PSK) and only for drivers that set WPA_DRIVER_FLAGS_4WAY_HANDSHAKE. This is the 8..63 character ASCII passphrase, if available. Please note that this can be NULL if passphrase was not used to generate the PSK. In that case, the psk field must be used to fetch the PSK.

14.380.2.8 const u8* wpa_driver_associate_params::prev_bssid

Previously used BSSID in this ESS. When not NULL, this is a request to use reassociation instead of association.

14.380.2.9 const u8* wpa_driver_associate_params::psk

RSN PSK (alternative for passphrase for PSK). This value is made available only for WPA/WPA2-Personal (PSK) and only for drivers that set WPA_DRIVER_FLAGS_4WAY_HANDSHAKE. This is the 32-octet (256-bit) PSK, if available. The driver wrapper should be prepared to handle NULL value as an error.

14.380.2.10 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 below).

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).

When using WPS, wpa_ie is used for WPS IE instead of WPA/RSN IE.

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

- src/drivers/driver.h

14.381 wpa_driver_atmel_data Struct Reference

Data Fields

- void * **wext**
- void * **ctx**
- char **ifname** [IFNAMSIZ+1]
- int **sock**

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

- src/drivers/driver_atmel.c

14.382 wpa_driver_auth_params Struct Reference

Authentication parameters.

`#include <driver.h>`

Data Fields

- int **freq**
- const u8 * **bssid**
- const u8 * **ssid**
- size_t **ssid_len**
- int **auth_alg**
- const u8 * **ie**
- size_t **ie_len**
- const u8 * **wep_key** [4]
- size_t **wep_key_len** [4]
- int **wep_tx_keyidx**

14.382.1 Detailed Description

Authentication parameters. Data for struct wpa_driver_ops::authenticate().

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

- src/drivers/driver.h

14.383 wpa_driver_broadcom_data Struct Reference

Data Fields

- void * **ctx**
- int **ioctl_sock**
- int **event_sock**
- char **ifname** [IFNAMSIZ+1]

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

- src/drivers/driver_broadcom.c

14.384 wpa_driver_bsd_data Struct Reference

Data Fields

- int **sock**
- int **route**
- char **ifname** [IFNAMSIZ+1]
- unsigned int **ifindex**
- void * **ctx**
- int **prev_roaming**
- int **prev_privacy**
- int **prev_wpa**

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

- src/drivers/driver_bsd.c

14.385 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**
- int **max_scan_ssids**

14.385.1 Detailed Description

Driver capability information.

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

- src/drivers/driver.h

14.386 wpa_driver_hostap_data Struct Reference

Data Fields

- void * **wext**
- void * **ctx**
- char **ifname** [IFNAMSIZ+1]
- int **sock**
- int **current_mode**

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

- src/drivers/driver_hostap.c

14.387 wpa_driver_iphone_data Struct Reference

Protected Attributes

- void * **ctx**
- Apple80211Ref **wireless_ctx**
- CFArrayRef **scan_results**
- int **ctrl_power**

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

- src/drivers/driver_iphone.m

14.388 wpa_driver_ipw_data Struct Reference

Data Fields

- void * **wext**
- void * **ctx**
- char **ifname** [IFNAMSIZ+1]
- int **sock**

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

- src/drivers/driver_ipw.c

14.389 wpa_driver_madwifi_data Struct Reference

Data Fields

- void * **wext**
- void * **ctx**
- char **ifname** [IFNAMSIZ+1]
- int **sock**

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

- src/drivers/driver_madwifi.c

14.390 wpa_driver_ndis_data Struct Reference

Data Fields

- void * **ctx**
- char **ifname** [100]
- u8 **own_addr** [ETH_ALEN]
- LPADAPTER **adapter**
- u8 **bssid** [ETH_ALEN]
- int **has_capability**
- int **no_of_pmkid**
- int **radio_enabled**
- struct wpa_driver_capa **capa**
- struct ndis_pmkid_entry * **pmkid**
- char * **adapter_desc**
- int **wired**
- int **native80211**
- int **mode**
- int **wzc_disabled**
- int **oid_bssid_set**

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

- src/drivers/driver_ndis.h

14.391 wpa_driver_ndiswrapper_data Struct Reference

Data Fields

- void * **wext**
- void * **ctx**
- char **ifname** [IFNAMSIZ+1]
- int **sock**

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

- src/drivers/driver_ndiswrapper.c

14.392 wpa_driver_nl80211_data Struct Reference

Data Fields

- void * **ctx**
- int **link_event_sock**
- int **ioctl_sock**
- char **ifname** [IFNAMSIZ+1]
- int **ifindex**
- int **if_removed**
- struct wpa_driver_capa **capa**
- int **has_capability**
- int **operstate**
- int **scan_complete_events**
- struct nl_handle * **nl_handle**
- struct nl_handle * **nl_handle_event**
- struct nl_cache * **nl_cache**
- struct nl_cache * **nl_cache_event**
- struct nl_cb * **nl_cb**
- struct genl_family * **nl80211**
- u8 **bssid** [ETH_ALEN]
- int **associated**
- u8 **ssid** [32]
- size_t **ssid_len**
- int **nlmode**
- int **ap_scan_as_station**
- int **monitor_sock**
- int **monitor_ifidx**
- unsigned int **beacon_set**:1

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

- src/drivers/driver_nl80211.c

14.393 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_key)(const char *ifname, 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(* 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(* 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)
  Get driver capabilities.
- void(* poll)(void *priv)
  Poll driver for association information.
- const char *(* get_ifname)(void *priv)
  Get interface name.
- const u8 *(* get_mac_addr)(void *priv)
  Get own MAC address.
- int(* send_eapol)(void *priv, const u8 *dest, u16 proto, const u8 *data, size_t data_len)
  Optional function for sending EAPOL packets.
- int(* set_operstate)(void *priv, int state)
  Sets device operating state to DORMANT or UP.
- int(* mlme_setprotection)(void *priv, const u8 *addr, int protect_type, int key_type)
  MLME-SETPROTECTION.request primitive.
- struct hostapd_hw_modes *(* get_hw_feature_data)(void *priv, u16 *num_modes, u16 *flags)
  Get hardware support data (channels and rates).
- int(* set_channel)(void *priv, hostapd_hw_mode phymode, int chan, int freq)
  Set channel.
- int(* set_ssid)(void *priv, const u8 *ssid, size_t ssid_len)
  Set SSID.
- int(* set_bssid)(void *priv, const u8 *bssid)
  Set BSSID.
- int(* send_mlme)(void *priv, const u8 *data, size_t data_len)
  Send management frame from MLME.
- int(* mlme_add_sta)(void *priv, const u8 *addr, const u8 *supp_rates, size_t supp_rates_len)
  Add a STA entry into the driver/netstack.
- int(* mlme_remove_sta)(void *priv, const u8 *addr)
  Remove a STA entry from the driver/netstack.
- int(* update_ft_ies)(void *priv, const u8 *md, const u8 *ies, size_t ies_len)
  Update FT (IEEE 802.11r) IEs.
- int(* send_ft_action)(void *priv, u8 action, const u8 *target_ap, const u8 *ies, size_t ies_len)
  Send FT Action frame (IEEE 802.11r).
- struct wpa_scan_results *(* get_scan_results2)(void *priv)
  Fetch the latest scan results.

- int(* set_country)(void *priv, const char *alpha2)
  Set country.
- void *(* global_init)(void)
  Global driver initialization.
- void(* global_deinit)(void *priv)
  Global driver deinitialization.
- void *(* init2)(void *ctx, const char *ifname, void *global_priv)
  Initialize driver interface (with global data).
- struct wpa_interface_info *(* get_interfaces)(void *global_priv)
  Get information about available interfaces.
- int(* scan2)(void *priv, struct wpa_driver_scan_params *params)
  Request the driver to initiate scan.
- int(* authenticate)(void *priv, struct wpa_driver_auth_params *params)
  Request driver to authenticate.
- int(* **set_beacon**)(const char *ifname, void *priv, const u8 *head, size_t head_len, const u8 *tail, size_t tail_len, int dtim_period, int beacon_int)
- void *(* **hapd_init**)(struct hostapd_data *hapd, struct wpa_init_params *params)
- void(* **hapd_deinit**)(void *priv)
- int(* set_ieee8021x)(const char *ifname, void *priv, int enabled)
  enable/disable IEEE 802.1X support
- int(* set_privacy)(const char *ifname, void *priv, int enabled)
  enable/disable privacy
- int(* **get_seqnum**)(const char *ifname, void *priv, const u8 *addr, int idx, u8 *seq)
- int(* **get_seqnum_igtk**)(const char *ifname, void *priv, const u8 *addr, int idx, u8 *seq)
- int(* **flush**)(void *priv)
- int(* **set_generic_elem**)(const char *ifname, void *priv, const u8 *elem, size_t elem_len)
- int(* **read_sta_data**)(void *priv, struct hostap_sta_driver_data *data, const u8 *addr)
- int(* **hapd_send_eapol**)(void *priv, const u8 *addr, const u8 *data, size_t data_len, int encrypt, const u8 *own_addr)
- int(* **sta_deauth**)(void *priv, const u8 *own_addr, const u8 *addr, int reason)
- int(* **sta_disassoc**)(void *priv, const u8 *own_addr, const u8 *addr, int reason)
- int(* **sta_remove**)(void *priv, const u8 *addr)
- int(* **hapd_get_ssid**)(const char *ifname, void *priv, u8 *buf, int len)
- int(* **hapd_set_ssid**)(const char *ifname, void *priv, const u8 *buf, int len)
- int(* **hapd_set_countermeasures**)(void *priv, int enabled)
- int(* **sta_add**)(const char *ifname, void *priv, struct hostapd_sta_add_params *params)
- int(* **get_inact_sec**)(void *priv, const u8 *addr)
- int(* **sta_clear_stats**)(void *priv, const u8 *addr)
- int(* **set_freq**)(void *priv, struct hostapd_freq_params *freq)
- int(* **set_rts**)(void *priv, int rts)

- int(∗ **set_frag**)(void ∗priv, int frag)
- int(∗ **sta_set_flags**)(void ∗priv, const u8 ∗addr, int total_flags, int flags_or, int flags_and)
- int(∗ **set_rate_sets**)(void ∗priv, int ∗supp_rates, int ∗basic_rates, int mode)
- int(∗ **set_internal_bridge**)(void ∗priv, int value)
- int(∗ **set_cts_protect**)(void ∗priv, int value)
- int(∗ **set_preamble**)(void ∗priv, int value)
- int(∗ **set_short_slot_time**)(void ∗priv, int value)
- int(∗ **set_tx_queue_params**)(void ∗priv, int queue, int aifs, int cw_min, int cw_max, int burst_time)

- int(∗ **bss_add**)(void ∗priv, const char ∗ifname, const u8 ∗bssid)
- int(∗ **bss_remove**)(void ∗priv, const char ∗ifname)
- int(∗ **valid_bss_mask**)(void ∗priv, const u8 ∗addr, const u8 ∗mask)
- int(∗ **if_add**)(const char ∗iface, void ∗priv, enum hostapd_driver_if_type type, char ∗ifname, const u8 ∗addr)
- int(∗ **if_update**)(void ∗priv, enum hostapd_driver_if_type type, char ∗ifname, const u8 ∗addr)
- int(∗ **if_remove**)(void ∗priv, enum hostapd_driver_if_type type, const char ∗ifname, const u8 ∗addr)

- int(∗ **set_sta_vlan**)(void ∗priv, const u8 ∗addr, const char ∗ifname, int vlan_id)
- int(∗ commit)(void ∗priv)

  Optional commit changes handler.

- int(∗ **send_ether**)(void ∗priv, const u8 ∗dst, const u8 ∗src, u16 proto, const u8 ∗data, size_t data_len)

- int(∗ **set_radius_acl_auth**)(void ∗priv, const u8 ∗mac, int accepted, u32 session_timeout)
- int(∗ **set_radius_acl_expire**)(void ∗priv, const u8 ∗mac)
- int(∗ **set_ht_params**)(const char ∗ifname, void ∗priv, const u8 ∗ht_capab, size_t ht_capab_len, const u8 ∗ht_oper, size_t ht_oper_len)
- int(∗ **set_wps_beacon_ie**)(const char ∗ifname, void ∗priv, const u8 ∗ie, size_t len)
- int(∗ **set_wps_probe_resp_ie**)(const char ∗ifname, void ∗priv, const u8 ∗ie, size_t len)
- int(∗ set_supp_port)(void ∗priv, int authorized)

  Set IEEE 802.1X Supplicant Port status.

14.393.1 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.

14.393.2 Field Documentation

14.393.2.1 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.

14.393.2.2 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

14.393.2.3 int(∗ wpa_driver_ops::authenticate)(void ∗priv, struct wpa_driver_auth_params ∗params)

Request driver to authenticate.

Parameters:

priv private driver interface data

params authentication parameters

Returns:

0 on success, -1 on failure

This is an optional function that can be used with drivers that support separate authentication and association steps, i.e., when wpa_supplicant can act as the SME. If not implemented, associate() function is expected to take care of IEEE 802.11 authentication, too.

14.393.2.4 int(∗ wpa_driver_ops::commit)(void ∗priv)

Optional commit changes handler.

Parameters:

priv driver private data

Returns:

0 on success, -1 on failure

This optional handler function can be registered if the driver interface implementation needs to commit changes (e.g., by setting network interface up) at the end of initial configuration. If set, this handler will be called after initial setup has been completed.

14.393.2.5 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

14.393.2.6 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.

14.393.2.7 const char∗ wpa_driver_ops::desc

One line description of the driver interface

14.393.2.8 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

14.393.2.9 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.

14.393.2.10 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.

14.393.2.11 int(∗ wpa_driver_ops::get_capa)(void ∗priv, struct wpa_driver_capa ∗capa)

Get driver capabilities.

Parameters:

priv private driver interface data

Returns:

0 on success, -1 on failure

Get driver/firmware/hardware capabilities.

14.393.2.12 struct hostapd_hw_modes∗(∗ wpa_driver_ops::get_hw_feature_data)(void ∗priv, u16 ∗num_modes, u16 ∗flags) [read]

Get hardware support data (channels and rates).

Parameters:

priv Private driver interface data

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

Returns:

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

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

14.393.2.13 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. Init() is called first.

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.

14.393.2.14 struct wpa_interface_info*(* wpa_driver_ops::get_interfaces)(void *global_priv) [read]

Get information about available interfaces.

Parameters:

global_priv private driver global data from global_init()

Returns:

Allocated buffer of interface information (caller is responsible for freeing the data structure) on success, NULL on failure

14.393.2.15 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.

14.393.2.16 struct wpa_scan_results*(* wpa_driver_ops::get_scan_results2)(void *priv) [read]

Fetch the latest scan results.

Parameters:

priv private driver interface data

Returns:

Allocated buffer of scan results (caller is responsible for freeing the data structure) on success, NULL on failure

14.393.2.17 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.

14.393.2.18 void(* wpa_driver_ops::global_deinit)(void *priv)

Global driver deinitialization.

Parameters:

priv private driver global data from global_init()

Terminate any global driver related functionality and free the global data structure.

14.393.2.19 void*(* wpa_driver_ops::global_init)(void)

Global driver initialization.

Returns:

Pointer to private data (global), NULL on failure

This optional function is called to initialize the driver wrapper for global data, i.e., data that applies to all interfaces. If this function is implemented, global_deinit() will also need to be implemented to free the private data. The driver will also likely use init2() function instead of init() to get the pointer to global data available to per-interface initializer.

14.393.2.20 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 below for more information about events and wpa_supplicant_event() function.

14.393.2.21 void*(* wpa_driver_ops::init2)(void *ctx, const char *ifname, void *global_priv)

Initialize driver interface (with global data).

Parameters:

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

ifname interface name, e.g., wlan0

global_priv private driver global data from global_init()

Returns:

Pointer to private data, NULL on failure

This function can be used instead of init() if the driver wrapper uses global data.

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

Add a STA entry into the driver/netstack.

Parameters:

priv Private driver interface data

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

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

supp_rates_len Number of entries in supp_rates

Returns:

0 on success, -1 on failure

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

14.393.2.23 int(∗ wpa_driver_ops::mlme_remove_sta)(void ∗priv, const u8 ∗addr)

Remove a STA entry from the driver/netstack.

Parameters:

priv Private driver interface data

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

Returns:

0 on success, -1 on failure

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

14.393.2.24 int(∗ wpa_driver_ops::mlme_setprotection)(void ∗priv, const u8 ∗addr, int protect_type, int key_type)

MLME-SETPROTECTION.request primitive.

Parameters:

priv Private driver interface data

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

protect_type MLME_SETPROTECTION_PROTECT_TYPE_∗

key_type MLME_SETPROTECTION_KEY_TYPE_∗

Returns:

0 on success, -1 on failure

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

14.393.2.25 const char* wpa_driver_ops::name

Name of the driver interface

14.393.2.26 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.

14.393.2.27 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.

14.393.2.28 int(* wpa_driver_ops::scan2)(void *priv, struct wpa_driver_scan_params *params)

Request the driver to initiate scan.

Parameters:

priv private driver interface data

params Scan parameters

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_results2().

14.393.2.29 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.

14.393.2.30 int(∗ wpa_driver_ops::send_ft_action)(void ∗priv, u8 action, const u8 ∗target_ap, const u8 ∗ies, size_t ies_len)

Send FT Action frame (IEEE 802.11r).

Parameters:

priv Private driver interface data

action Action field value

target_ap Target AP address

ies FT IEs (MDIE, FTIE, ...) (FT Request action frame body)

ies_len Length of FT IEs in bytes

Returns:

0 on success, -1 on failure

The supplicant uses this callback to request the driver to transmit an FT Action frame (action category 6) for over-the-DS fast BSS transition.

14.393.2.31 int(∗ wpa_driver_ops::send_mlme)(void ∗priv, const u8 ∗data, size_t data_len)

Send management frame from MLME.

Parameters:

priv Private driver interface data

data IEEE 802.11 management frame with IEEE 802.11 header

data_len Size of the management frame

Returns:

0 on success, -1 on failure

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

14.393.2.32 int(∗ wpa_driver_ops::set_bssid)(void ∗priv, const u8 ∗bssid)

Set BSSID.

Parameters:

priv Private driver interface data

bssid BSSID

Returns:

0 on success, -1 on failure

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

14.393.2.33 int(∗ wpa_driver_ops::set_channel)(void ∗priv, hostapd_hw_mode phymode, int chan, int freq)

Set channel.

Parameters:

priv Private driver interface data

phymode HOSTAPD_MODE_IEEE80211B, ..

chan IEEE 802.11 channel number

freq Frequency of the channel in MHz

Returns:

0 on success, -1 on failure

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

14.393.2.34 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.

14.393.2.35 int(∗ wpa_driver_ops::set_country)(void ∗priv, const char ∗alpha2)

Set country.

Parameters:

priv Private driver interface data

alpha2 country to which to switch to

Returns:

0 on success, -1 on failure

This function is for drivers which support some form of setting a regulatory domain.

14.393.2.36 int(∗ wpa_driver_ops::set_ieee8021x)(const char ∗ifname, void ∗priv, int enabled)

enable/disable IEEE 802.1X support

Parameters:

ifname Interface name (for multi-SSID/VLAN support)

priv driver private data

enabled 1 = enable, 0 = disable

Returns:

0 on success, -1 on failure

Configure the kernel driver to enable/disable 802.1X support. This may be an empty function if 802.1X support is always enabled.

14.393.2.37 int(∗ wpa_driver_ops::set_key)(const char ∗ifname, 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:

ifname Interface name (for multi-SSID/VLAN support)

priv private driver interface data

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

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

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

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

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

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

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

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

Returns:

0 on success, -1 on failure

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

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

14.393.2.38 int(∗ wpa_driver_ops::set_operstate)(void ∗priv, int state)

Sets device operating state to DORMANT or UP.

Parameters:

priv private driver interface data

state 0 = dormant, 1 = up

Returns:

0 on success, -1 on failure

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

14.393.2.39 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).

14.393.2.40 int(∗ wpa_driver_ops::set_privacy)(const char ∗ifname, void ∗priv, int enabled)

enable/disable privacy

Parameters:

priv driver private data

enabled 1 = privacy enabled, 0 = disabled

Return: 0 on success, -1 on failure

Configure privacy.

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

Set SSID.

Parameters:

priv Private driver interface data

ssid SSID

ssid_len SSID length

Returns:

0 on success, -1 on failure

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

14.393.2.42 int(∗ wpa_driver_ops::set_supp_port)(void ∗priv, int authorized)

Set IEEE 802.1X Supplicant Port status.

Parameters:

priv Private driver interface data

authorized Whether the port is authorized

Returns:

0 on success, -1 on failure

14.393.2.43 int(∗ wpa_driver_ops::update_ft_ies)(void ∗priv, const u8 ∗md, const u8 ∗ies, size_t ies_len)

Update FT (IEEE 802.11r) IEs.

Parameters:

priv Private driver interface data

md Mobility domain (2 octets) (also included inside ies)

ies FT IEs (MDIE, FTIE, ...) or NULL to remove IEs

ies_len Length of FT IEs in bytes

Returns:

0 on success, -1 on failure

The supplicant uses this callback to let the driver know that keying material for FT is available and that the driver can use the provided IEs in the next message in FT authentication sequence.

This function is only needed for driver that support IEEE 802.11r (Fast BSS Transition).

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

- src/drivers/driver.h

14.394 wpa_driver_osx_data Struct Reference

Protected Attributes

- void * **ctx**
- WirelessRef **wireless_ctx**
- CFArrayRef **scan_results**

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

- src/drivers/driver_osx.m

14.395 wpa_driver_prism54_data Struct Reference

Data Fields

- void * **wext**
- void * **ctx**
- char **ifname** [IFNAMSIZ+1]
- int **sock**

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

- src/drivers/driver_prism54.c

14.396 wpa_driver_privsep_data Struct Reference

Data Fields

- void * **ctx**
- u8 **own_addr** [ETH_ALEN]
- int **priv_socket**
- char * **own_socket_path**
- int **cmd_socket**
- char * **own_cmd_path**
- struct sockaddr_un **priv_addr**
- char **ifname** [16]

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

- src/drivers/driver_privsep.c

14.397 wpa_driver_ralink_data Struct Reference

Data Fields

- void * **ctx**
- int **ioctl_sock**
- int **event_sock**
- char **ifname** [IFNAMSIZ+1]
- u8 * **assoc_req_ies**
- size_t **assoc_req_ies_len**
- u8 * **assoc_resp_ies**
- size_t **assoc_resp_ies_len**
- int **no_of_pmkid**
- struct ndis_pmkid_entry * **pmkid**
- int **we_version_compiled**
- int **ap_scan**
- int **scanning_done**
- u8 **g_driver_down**

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

- src/drivers/driver_ralink.c

14.398 wpa_driver_roboswitch_data Struct Reference

Data Fields

- void * **ctx**
- struct l2_packet_data * **l2**
- char **ifname** [IFNAMSIZ+1]
- u8 **own_addr** [ETH_ALEN]
- struct ifreq **ifr**
- int **fd**
- int **is_5350**
- u16 **ports**

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

- src/drivers/driver_roboswitch.c

14.399 wpa_driver_scan_params Struct Reference

Scan parameters.

`#include <driver.h>`

Data Structures

- struct wpa_driver_scan_ssid
  SSIDs to scan for.

Data Fields

- struct wpa_driver_scan_params::wpa_driver_scan_ssid ssids [WPAS_MAX_SCAN_SSIDS]
  SSIDs to scan for.
- size_t num_ssids
  Number of entries in ssids array.
- const u8 * extra_ies
  Extra IE(s) to add into Probe Request or NULL.
- size_t extra_ies_len
  Length of extra_ies in octets.
- int * freqs
  Array of frequencies to scan or NULL for all frequencies.

14.399.1 Detailed Description

Scan parameters. Data for struct wpa_driver_ops::scan2().

14.399.2 Field Documentation

14.399.2.1 int* wpa_driver_scan_params::freqs

Array of frequencies to scan or NULL for all frequencies. The frequency is set in MHz. The array is zero-terminated.

14.399.2.2 size_t wpa_driver_scan_params::num_ssids

Number of entries in ssids array. Zero indicates a request for a passive scan.

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

- src/drivers/driver.h

14.400 wpa_driver_scan_params::wpa_driver_scan_ssid Struct Reference

SSIDs to scan for.

`#include <driver.h>`

Data Fields

- const u8 * ssid

  specific SSID to scan for (ProbeReq)

- size_t ssid_len

14.400.1 Detailed Description

SSIDs to scan for.

14.400.2 Field Documentation

14.400.2.1 const u8* wpa_driver_scan_params::wpa_driver_scan_ssid::ssid

specific SSID to scan for (ProbeReq) NULL or zero-length SSID is used to indicate active scan with wildcard SSID.

14.400.2.2 size_t wpa_driver_scan_params::wpa_driver_scan_ssid::ssid_len

ssid_len: Length of the SSID in octets

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

- src/drivers/driver.h

14.401 wpa_driver_test_data Struct Reference

Data Fields

- struct wpa_driver_test_global * **global**
- void * **ctx**
- u8 **own_addr** [ETH_ALEN]
- int **test_socket**
- struct sockaddr_un **hostapd_addr**
- int **hostapd_addr_set**
- struct sockaddr_in **hostapd_addr_udp**
- int **hostapd_addr_udp_set**
- char * **own_socket_path**
- char * **test_dir**
- u8 **bssid** [ETH_ALEN]
- u8 **ssid** [32]
- size_t **ssid_len**
- struct wpa_scan_res * **scanres** [MAX_SCAN_RESULTS]
- size_t **num_scanres**
- int **use_associnfo**
- u8 **assoc_wpa_ie** [80]
- size_t **assoc_wpa_ie_len**
- int **use_mlme**
- int **associated**
- u8 * **probe_req_ie**
- size_t **probe_req_ie_len**
- int **ibss**
- int **privacy**
- int **ap**
- struct hostapd_data * **hapd**
- struct test_client_socket * **cli**
- struct test_driver_bss * **bss**
- int **udp_port**

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

- src/drivers/driver_test.c

14.402 wpa_driver_test_global Struct Reference

Data Fields

- int **dummy**

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

- src/drivers/driver_test.c

14.403 wpa_driver_wext_data Struct Reference

Data Fields

- void * **ctx**
- int **event_sock**
- int **ioctl_sock**
- int **mlme_sock**
- char **ifname** [IFNAMSIZ+1]
- int **ifindex**
- int **ifindex2**
- int **if_removed**
- u8 * **assoc_req_ies**
- size_t **assoc_req_ies_len**
- u8 * **assoc_resp_ies**
- size_t **assoc_resp_ies_len**
- struct wpa_driver_capa **capa**
- int **has_capability**
- int **we_version_compiled**
- int **use_crypt**
- int **auth_alg_fallback**
- int **operstate**
- char **mlmedev** [IFNAMSIZ+1]
- int **scan_complete_events**

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

- src/drivers/driver_wext.h

14.404 wpa_driver_wired_data Struct Reference

Data Fields

- void * **ctx**
- int **pf_sock**
- char **ifname** [IFNAMSIZ+1]
- int **membership**
- int **multi**
- int **iff_allmulti**
- int **iff_up**

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

- src/drivers/driver_wired.c

14.405 wpa_eapol_ie_parse Struct Reference

Data Fields

- const u8 * **wpa_ie**
- size_t **wpa_ie_len**
- const u8 * **rsn_ie**
- size_t **rsn_ie_len**
- const u8 * **pmkid**
- const u8 * **gtk**
- size_t **gtk_len**
- const u8 * **mac_addr**
- size_t **mac_addr_len**

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

- hostapd/wpa_auth_ie.h
- src/rsn_supp/wpa_ie.h

14.406 wpa_eapol_key Struct Reference

Data Fields

- u8 **type**
- u8 **key_info** [2]
- u8 **key_length** [2]
- u8 **replay_counter** [WPA_REPLAY_COUNTER_LEN]
- u8 **key_nonce** [WPA_NONCE_LEN]
- u8 **key_iv** [16]
- u8 **key_rsc** [WPA_KEY_RSC_LEN]
- u8 **key_id** [8]
- u8 **key_mic** [16]
- u8 **key_data_length** [2]

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

- src/common/wpa_common.h

14.407 wpa_event_data Union Reference

`#include <driver.h>`

Data Structures

- struct assoc_info
  Data for EVENT_ASSOC and EVENT_ASSOCINFO events.
- struct assoc_reject
  Data for EVENT_ASSOC_REJECT events.
- struct auth_info
  Data for EVENT_AUTH events.
- struct ft_ies
  FT information elements (EVENT_FT_RESPONSE).
- struct ibss_rsn_start
  Data for EVENT_IBSS_RSN_START.
- struct interface_status
  Data for EVENT_INTERFACE_STATUS.
- struct michael_mic_failure
  Data for EVENT_MICHAEL_MIC_FAILURE.
- struct pmkid_candidate
  Data for EVENT_PMKID_CANDIDATE.
- struct stkstart
  Data for EVENT_STKSTART.
- struct timeout_event

Data Fields

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

- struct wpa_event_data::stkstart stkstart
  Data for EVENT_STKSTART.
- struct wpa_event_data::ft_ies ft_ies
  FT information elements (EVENT_FT_RESPONSE).
- struct wpa_event_data::ibss_rsn_start ibss_rsn_start
  Data for EVENT_IBSS_RSN_START.
- struct wpa_event_data::auth_info auth
  Data for EVENT_AUTH events.
- struct wpa_event_data::assoc_reject assoc_reject
  Data for EVENT_ASSOC_REJECT events.
- struct wpa_event_data::timeout_event **timeout_event**

14.407.1 Detailed Description

union wpa_event_data - Additional data for wpa_supplicant_event() calls

14.407.2 Field Documentation

14.407.2.1 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.

14.407.2.2 struct wpa_event_data::ft_ies wpa_event_data::ft_ies

FT information elements (EVENT_FT_RESPONSE). During FT (IEEE 802.11r) authentication sequence, the driver is expected to use this event to report received FT IEs (MDIE, FTIE, RSN IE, TIE, possible resource request) to the supplicant. The FT IEs for the next message will be delivered through the struct wpa_driver_ops::update_ft_ies() callback.

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

- src/drivers/driver.h

14.408 wpa_global Struct Reference

Internal, global data for all wpa_supplicant interfaces.

`#include <wpa_supplicant_i.h>`

Data Fields

- struct wpa_supplicant * **ifaces**
- struct wpa_params **params**
- struct ctrl_iface_global_priv * **ctrl_iface**
- struct ctrl_iface_dbus_priv * **dbus_ctrl_iface**
- struct ctrl_iface_dbus_new_priv * **dbus_new_ctrl_iface**
- void ** **drv_priv**
- size_t **drv_count**

14.408.1 Detailed Description

Internal, global data for all wpa_supplicant interfaces. This structure is initialized by calling wpa_supplicant_init() when starting wpa_supplicant.

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

- wpa_supplicant/wpa_supplicant_i.h

14.409 wpa_global_dst Struct Reference

Data Fields

- OVERLAPPED **overlap**
- struct wpa_global_dst * **next**
- struct wpa_global_dst * **prev**
- struct ctrl_iface_global_priv * **priv**
- HANDLE **pipe**
- char **req_buf** [REQUEST_BUFSIZE]
- char * **rsp_buf**
- int **used**

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

- wpa_supplicant/ctrl_iface_named_pipe.c

14.410 wpa_group Struct Reference

Public Types

- enum { **WPA_GROUP_GTK_INIT** = 0, **WPA_GROUP_SETKEYS**, **WPA_GROUP_-SETKEYSDONE** }

Data Fields

- struct wpa_group * **next**
- int **vlan_id**
- Boolean **GInit**
- int **GKeyDoneStations**
- Boolean **GTKReKey**
- int **GTK_len**
- int **GN**
- int **GM**
- Boolean **GTKAuthenticator**
- u8 **Counter** [WPA_NONCE_LEN]
- enum wpa_group:: { ... } **wpa_group_state**
- u8 **GMK** [WPA_GMK_LEN]
- u8 **GTK** [2][WPA_GTK_MAX_LEN]
- u8 **GNonce** [WPA_NONCE_LEN]
- Boolean **changed**

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

- hostapd/wpa_auth_i.h

14.411 wpa_gtk_data Struct Reference

Data Fields

- wpa_alg **alg**
- int **tx**
- int **key_rsc_len**
- int **keyidx**
- u8 **gtk** [32]
- int **gtk_len**

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

- src/rsn_supp/wpa.c

14.412 wpa_ie_data Struct Reference

Data Fields

- int **proto**
- int **pairwise_cipher**
- int **group_cipher**
- int **key_mgmt**
- int **capabilities**
- size_t **num_pmkid**
- const u8 * **pmkid**
- int **mgmt_group_cipher**

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

- src/common/wpa_common.h

14.413 wpa_ie_hdr Struct Reference

Data Fields

- u8 **elem_id**
- u8 **len**
- u8 **oui** [4]
- u8 **version** [2]

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

- src/common/wpa_common.h

14.414 wpa_init_params Struct Reference

Data Fields

- const u8 * **bssid**
- const char * **ifname**
- const u8 * **ssid**
- size_t **ssid_len**
- const char * **test_socket**
- int **use_pae_group_addr**
- char ** **bridge**
- size_t **num_bridge**
- u8 * **own_addr**

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

- src/drivers/driver.h

14.415 wpa_interface Struct Reference

Parameters for wpa_supplicant_add_iface().

`#include <wpa_supplicant_i.h>`

Data Fields

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

14.415.1 Detailed Description

Parameters for wpa_supplicant_add_iface().

14.415.2 Field Documentation

14.415.2.1 const char* wpa_interface::bridge_ifname

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

14.415.2.2 const char* wpa_interface::confname

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

14.415.2.3 const char* wpa_interface::ctrl_interface

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

14.415.2.4 const char* wpa_interface::driver_param

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

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

- wpa_supplicant/wpa_supplicant_i.h

14.416 wpa_interface_info Struct Reference

Network interface information.

`#include <driver.h>`

Data Fields

- struct wpa_interface_info * **next**
- char * **ifname**
- char * **desc**
- const char * **drv_name**

14.416.1 Detailed Description

Network interface information.

Parameters:

next Pointer to the next interface or NULL if this is the last one

ifname Interface name that can be used with init() or init2()

desc Human readable adapter description (e.g., vendor/model) or NULL if not available

drv_bame struct wpa_driver_ops::name (note: unlike other strings, this one is not an allocated copy, i.e., get_interfaces() caller will not free this)

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

- src/drivers/driver.h

14.417 wpa_key Struct Reference

Data Fields

- wpa_alg **alg**
- const u8 ∗ **addr**
- int **key_index**
- int **set_tx**
- const u8 ∗ **seq**
- size_t **seq_len**
- const u8 ∗ **key**
- size_t **key_len**

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

- src/drivers/driver_ndiswrapper.c

14.418 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_monitor

  Wait for a monitor program before starting.

- char * pid_file

  Path to a PID (process ID) file.

- int wpa_debug_level

  Debugging verbosity level (e.g., MSG_INFO).

- int wpa_debug_show_keys

  Whether keying material is included in debug.

- int wpa_debug_timestamp

  Whether to include timestamp in debug messages.

- char * ctrl_interface

  Global ctrl_iface path/parameter.

- int dbus_ctrl_interface

  Enable the DBus control interface.

- const char * wpa_debug_file_path

  Path of debug file or NULL to use stdout.

- int wpa_debug_syslog

  Enable log output through syslog.

- char * override_driver

  Optional driver parameter override.

- char * override_ctrl_interface

  Optional ctrl_interface override.

14.418.1 Detailed Description

Parameters for wpa_supplicant_init().

14.418.2 Field Documentation

14.418.2.1 char* wpa_params::override_ctrl_interface

Optional ctrl_interface override. This parameter can be used to override the ctrl_interface parameter in dynamic interface addition to force a control interface to be created.

14.418.2.2 char* wpa_params::override_driver

Optional driver parameter override. This parameter can be used to override the driver parameter in dynamic interface addition to force a specific driver wrapper to be used instead.

14.418.2.3 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.

14.418.2.4 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.

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

- wpa_supplicant/wpa_supplicant_i.h

14.419 wpa_peerkey Struct Reference

Data Fields

- struct wpa_peerkey * **next**
- int **initiator**
- u8 **addr** [ETH_ALEN]
- u8 **inonce** [WPA_NONCE_LEN]
- u8 **pnonce** [WPA_NONCE_LEN]
- u8 **rsnie_i** [PEERKEY_MAX_IE_LEN]
- size_t **rsnie_i_len**
- u8 **rsnie_p** [PEERKEY_MAX_IE_LEN]
- size_t **rsnie_p_len**
- u8 **smk** [PMK_LEN]
- int **smk_complete**
- u8 **smkid** [PMKID_LEN]
- u32 **lifetime**
- os_time_t **expiration**
- int **cipher**
- u8 **replay_counter** [WPA_REPLAY_COUNTER_LEN]
- int **replay_counter_set**
- int **use_sha256**
- struct wpa_ptk stk **tstk**
- int **stk_set**
- int **tstk_set**

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

- src/rsn_supp/peerkey.h

14.420 wpa_priv_interface Struct Reference

Data Fields

- struct wpa_priv_interface * **next**
- char * **driver_name**
- char * **ifname**
- char * **sock_name**
- int **fd**
- struct wpa_driver_ops * **driver**
- void * **drv_priv**
- struct sockaddr_un **drv_addr**
- int **wpas_registered**
- struct l2_packet_data * **l2**
- struct sockaddr_un **l2_addr**

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

- wpa_supplicant/wpa_priv.c

14.421 wpa_ptk Struct Reference

WPA Pairwise Transient Key.

`#include <wpa_common.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**

14.421.1 Detailed Description

WPA Pairwise Transient Key. IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy

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

- src/common/wpa_common.h

14.422 wpa_scan_res Struct Reference

Scan result for an BSS/IBSS.

`#include <driver.h>`

Data Fields

- unsigned int **flags**
- u8 **bssid** [ETH_ALEN]
- int **freq**
- u16 **beacon_int**
- u16 **caps**
- int **qual**
- int **noise**
- int **level**
- u64 **tsf**
- unsigned int **age**
- size_t **ie_len**

14.422.1 Detailed Description

Scan result for an BSS/IBSS.

Parameters:

flags information flags about the BSS/IBSS (WPA_SCAN_∗)

bssid BSSID

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

beacon_int beacon interval in TUs (host byte order)

caps capability information field in host byte order

qual signal quality

noise noise level

level signal level

tsf Timestamp

age Age of the information in milliseconds (i.e., how many milliseconds ago the last Beacon or Probe Response frame was received)

ie_len length of the following IE field in octets

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.

If the driver does not support reporting all IEs, the IE data structure is constructed of the IEs that are available. This field will also need to include SSID in IE format. All drivers are encouraged to be extended to report all IEs to make it easier to support future additions.

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

- src/drivers/driver.h

14.423 wpa_scan_results Struct Reference

Scan results.

`#include <driver.h>`

Data Fields

- struct wpa_scan_res ** **res**
- size_t **num**

14.423.1 Detailed Description

Scan results.

Parameters:

res Array of pointers to allocated variable length scan result entries

num Number of entries in the scan result array

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

- src/drivers/driver.h

14.424 wpa_sm Struct Reference

Internal WPA state machine data.

`#include <wpa_i.h>`

Data Fields

- u8 **pmk** [PMK_LEN]
- size_t **pmk_len**
- struct wpa_ptk ptk **tptk**
- int **ptk_set**
- int **tptk_set**
- u8 **snonce** [WPA_NONCE_LEN]
- u8 **anonce** [WPA_NONCE_LEN]
- int **renew_snonce**
- u8 **rx_replay_counter** [WPA_REPLAY_COUNTER_LEN]
- int **rx_replay_counter_set**
- u8 **request_counter** [WPA_REPLAY_COUNTER_LEN]
- struct eapol_sm * **eapol**
- struct rsn_pmksa_cache * **pmksa**
- struct rsn_pmksa_cache_entry * **cur_pmksa**
- struct rsn_pmksa_candidate * **pmksa_candidates**
- struct l2_packet_data * **l2_preauth**
- struct l2_packet_data * **l2_preauth_br**
- u8 **preauth_bssid** [ETH_ALEN]
- struct eapol_sm * **preauth_eapol**
- struct wpa_sm_ctx * **ctx**
- void * **scard_ctx**
- int **fast_reauth**
- void * **network_ctx**
- int **peerkey_enabled**
- int **allowed_pairwise_cipher**
- int **proactive_key_caching**
- int **eap_workaround**
- void * **eap_conf_ctx**
- u8 **ssid** [32]
- size_t **ssid_len**
- int **wpa_ptk_rekey**
- u8 **own_addr** [ETH_ALEN]
- const char * **ifname**
- const char * **bridge_ifname**
- u8 **bssid** [ETH_ALEN]
- unsigned int **dot11RSNAConfigPMKLifetime**
- unsigned int **dot11RSNAConfigPMKReauthThreshold**
- unsigned int **dot11RSNAConfigSATimeout**
- unsigned int **dot11RSNA4WayHandshakeFailures**
- unsigned int **proto**
- unsigned int **pairwise_cipher**
- unsigned int **group_cipher**

- unsigned int **key_mgmt**
- unsigned int **mgmt_group_cipher**
- int **rsn_enabled**
- 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**

14.424.1 Detailed Description

Internal WPA state machine data.

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

- src/rsn_supp/wpa_i.h

14.425 wpa_sm_ctx Struct Reference

Data Fields

- void ∗ **ctx**
- void ∗ **msg_ctx**
- void(∗ **set_state**)(void ∗ctx, wpa_states state)
- wpa_states(∗ **get_state**)(void ∗ctx)
- void(∗ **deauthenticate**)(void ∗ctx, int reason_code)
- void(∗ **disassociate**)(void ∗ctx, int reason_code)
- int(∗ **set_key**)(void ∗ctx, 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)
- void ∗(∗ **get_network_ctx**)(void ∗ctx)
- int(∗ **get_bssid**)(void ∗ctx, u8 ∗bssid)
- int(∗ **ether_send**)(void ∗ctx, const u8 ∗dest, u16 proto, const u8 ∗buf, size_t len)
- int(∗ **get_beacon_ie**)(void ∗ctx)
- void(∗ **cancel_auth_timeout**)(void ∗ctx)
- u8 ∗(∗ **alloc_eapol**)(void ∗ctx, u8 type, const void ∗data, u16 data_len, size_t ∗msg_len, void ∗∗data_pos)
- int(∗ **add_pmkid**)(void ∗ctx, const u8 ∗bssid, const u8 ∗pmkid)
- int(∗ **remove_pmkid**)(void ∗ctx, const u8 ∗bssid, const u8 ∗pmkid)
- void(∗ **set_config_blob**)(void ∗ctx, struct wpa_config_blob ∗blob)
- struct wpa_config_blob ∗(∗ **get_config_blob**)(void ∗ctx, const char ∗name)
- int(∗ **mlme_setprotection**)(void ∗ctx, const u8 ∗addr, int protection_type, int key_type)
- int(∗ **update_ft_ies**)(void ∗ctx, const u8 ∗md, const u8 ∗ies, size_t ies_len)
- int(∗ **send_ft_action**)(void ∗ctx, u8 action, const u8 ∗target_ap, const u8 ∗ies, size_t ies_len)

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

- src/rsn_supp/wpa.h

14.426 wpa_ssid Struct Reference

Network configuration data.

`#include <config_ssid.h>`

Data Fields

- struct wpa_ssid * next
  Next network in global list.
- struct 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 [32]
  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.
- int eapol_flags
  Bit field of IEEE 802.1X/EAPOL options (EAPOL_FLAG_∗).
- struct eap_peer_config eap
  EAP peer configuration for this network.
- 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.
- int mixed_cell
  Whether mixed cells are allowed.
- 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.
- int mode
  IEEE 802.11 operation mode (Infrastucture/IBSS).
- int disabled
  Whether this network is currently disabled.
- int peerkey
  Whether PeerKey handshake for direct links is allowed.
- char ∗ id_str
  Network identifier string for external scripts.

- int frequency

  Channel frequency in megahertz (MHz) for IBSS.

- int wpa_ptk_rekey

  Maximum lifetime for PTK in seconds.

- int * scan_freq

  Array of frequencies to scan or NULL for all.

- char * bgscan

  Background scan and roaming parameters or NULL if none.

14.426.1 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.

14.426.2 Field Documentation

14.426.2.1 int wpa_ssid::auth_alg

Bitfield of allowed authentication algorithms. WPA_AUTH_ALG_*

14.426.2.2 char* wpa_ssid::bgscan

Background scan and roaming parameters or NULL if none. This is an optional set of parameters for background scanning and roaming within a network (ESS) in following format: <bgscan module="" name>="">:<module parameters>="">

14.426.2.3 u8 wpa_ssid::bssid[ETH_ALEN]

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

14.426.2.4 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).

14.426.2.5 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.

14.426.2.6 int wpa_ssid::frequency

Channel frequency in megahertz (MHz) for IBSS. This value is used to configure the initial channel for IBSS (adhoc) networks, e.g., 2412 = IEEE 802.11b/g channel 1. It is ignored in the infrastructure mode. In addition, this value is only used by the station that creates the IBSS. If an IBSS network with the configured SSID is already present, the frequency of the network will be used instead of this configured value.

14.426.2.7 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.

14.426.2.8 char* wpa_ssid::id_str

Network identifier string for external scripts. This value is passed to external ctrl_iface monitors in WPA_-EVENT_CONNECTED event and wpa_cli sets this as WPA_ID_STR environment variable for action scripts.

14.426.2.9 int wpa_ssid::key_mgmt

Bitfield of allowed key management protocols. WPA_KEY_MGMT_*

14.426.2.10 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.

14.426.2.11 int wpa_ssid::mixed_cell

Whether mixed cells are allowed. This option can be used to configure whether so called mixed cells, i.e., networks that use both plaintext and encryption in the same SSID, are allowed. This is disabled (0) by default. Enable by setting this to 1.

14.426.2.12 int wpa_ssid::mode

IEEE 802.11 operation mode (Infrastucture/IBSS). 0 = infrastructure (Managed) mode, i.e., associate with an AP.

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

2 = AP (access point)

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).

14.426.2.13 struct wpa_ssid* wpa_ssid::next [read]

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.

14.426.2.14 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.

14.426.2.15 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).

14.426.2.16 int wpa_ssid::peerkey

Whether PeerKey handshake for direct links is allowed. This is only used when both RSN/WPA2 and IEEE 802.11e (QoS) are enabled.

0 = disabled (default) 1 = enabled

14.426.2.17 struct wpa_ssid* wpa_ssid::pnext [read]

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.

14.426.2.18 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.

14.426.2.19 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.

14.426.2.20 int* wpa_ssid::scan_freq

Array of frequencies to scan or NULL for all. This is an optional zero-terminated array of frequencies in megahertz (MHz) to include in scan requests when searching for this network. This can be used to speed up scanning when the network is known to not use all possible channels.

14.426.2.21 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.

14.426.2.22 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.

14.426.2.23 int wpa_ssid::wpa_ptk_rekey

Maximum lifetime for PTK in seconds. This value can be used to enforce rekeying of PTK to mitigate some attacks against TKIP deficiencies.

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

- wpa_supplicant/config_ssid.h

14.427 wpa_state_machine Struct Reference

Public Types

- enum {

  WPA_PTK_INITIALIZE, **WPA_PTK_DISCONNECT**, **WPA_PTK_DISCONNECTED**, **WPA_PTK_AUTHENTICATION**,

  WPA_PTK_AUTHENTICATION2, **WPA_PTK_INITPMK**, **WPA_PTK_INITPSK**, **WPA_PTK_PTKSTART**,

  WPA_PTK_PTKCALCNEGOTIATING, **WPA_PTK_PTKCALCNEGOTIATING2**, **WPA_PTK_PTKINITNEGOTIATING**, **WPA_PTK_PTKINITDONE** }
- enum { **WPA_PTK_GROUP_IDLE** = 0, **WPA_PTK_GROUP_REKEYNEGOTIATING**, **WPA_PTK_GROUP_REKEYESTABLISHED**, **WPA_PTK_GROUP_KEYERROR** }
- enum { **WPA_VERSION_NO_WPA** = 0, **WPA_VERSION_WPA** = 1, **WPA_VERSION_WPA2** = 2 }

Data Fields

- struct wpa_authenticator * **wpa_auth**
- struct wpa_group * **group**
- u8 **addr** [ETH_ALEN]
- enum wpa_state_machine:: { ... } **wpa_ptk_state**
- enum wpa_state_machine:: { ... } **wpa_ptk_group_state**
- Boolean **Init**
- Boolean **DeauthenticationRequest**
- Boolean **AuthenticationRequest**
- Boolean **ReAuthenticationRequest**
- Boolean **Disconnect**
- int **TimeoutCtr**
- int **GTimeoutCtr**
- Boolean **TimeoutEvt**
- Boolean **EAPOLKeyReceived**
- Boolean **EAPOLKeyPairwise**
- Boolean **EAPOLKeyRequest**
- Boolean **MICVerified**
- Boolean **GUpdateStationKeys**
- u8 **ANonce** [WPA_NONCE_LEN]
- u8 **SNonce** [WPA_NONCE_LEN]
- u8 **PMK** [PMK_LEN]
- struct wpa_ptk **PTK**
- Boolean **PTK_valid**
- Boolean **pairwise_set**
- int **keycount**
- Boolean **Pair**
- struct {
  u8 **counter** [WPA_REPLAY_COUNTER_LEN]
  Boolean **valid**
  } **key_replay** [RSNA_MAX_EAPOL_RETRIES]

- Boolean **PInitAKeys**
- Boolean **PTKRequest**
- Boolean **has_GTK**
- Boolean **PtkGroupInit**
- u8 * **last_rx_eapol_key**
- size_t **last_rx_eapol_key_len**
- unsigned int **changed**:1
- unsigned int **in_step_loop**:1
- unsigned int **pending_deinit**:1
- unsigned int **started**:1
- unsigned int **mgmt_frame_prot**:1
- u8 **req_replay_counter** [WPA_REPLAY_COUNTER_LEN]
- int **req_replay_counter_used**
- u8 * **wpa_ie**
- size_t **wpa_ie_len**
- enum wpa_state_machine:: { ... } **wpa**
- int **pairwise**
- int **wpa_key_mgmt**
- struct rsn_pmksa_cache_entry * **pmksa**
- u32 **dot11RSNAStatsTKIPLocalMICFailures**
- u32 **dot11RSNAStatsTKIPRemoteMICFailures**

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

- hostapd/wpa_auth_i.h

14.428 wpa_stsl_negotiation Struct Reference

Data Fields

- struct wpa_stsl_negotiation * **next**
- u8 **initiator** [ETH_ALEN]
- u8 **peer** [ETH_ALEN]

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

- hostapd/wpa_auth_i.h

14.429 wpa_supplicant Struct Reference

Internal data for wpa_supplicant interface.

`#include <wpa_supplicant_i.h>`

Data Fields

- struct wpa_global * **global**
- struct wpa_supplicant * **next**
- struct l2_packet_data * **l2**
- struct l2_packet_data * **l2_br**
- unsigned char **own_addr** [ETH_ALEN]
- char **ifname** [100]
- char **bridge_ifname** [16]
- char * **confname**
- struct wpa_config * **conf**
- int **countermeasures**
- os_time_t **last_michael_mic_error**
- u8 **bssid** [ETH_ALEN]
- u8 **pending_bssid** [ETH_ALEN]
- int **reassociate**
- int **disconnected**
- struct wpa_ssid * **current_ssid**
- int **ap_ies_from_associnfo**
- int **pairwise_cipher**
- int **group_cipher**
- int **key_mgmt**
- int **mgmt_group_cipher**
- void * **drv_priv**
- void * **global_drv_priv**
- struct wpa_ssid * **prev_scan_ssid**
- struct wpa_scan_results * **scan_res**
- struct wpa_driver_ops * **driver**
- int **interface_removed**
- struct wpa_sm * **wpa**
- struct eapol_sm * **eapol**
- struct ctrl_iface_priv * **ctrl_iface**
- wpa_states **wpa_state**
- int **scanning**
- int **new_connection**
- int **reassociated_connection**
- int **eapol_received**
- struct scard_data * **scard**
- unsigned char **last_eapol_src** [ETH_ALEN]
- int **keys_cleared**
- struct wpa_blacklist * **blacklist**
- int **scan_req**
- int **scan_res_tried**
- int **scan_runs**

- struct wpa_client_mlme **mlme**
- unsigned int **drv_flags**
- int **max_scan_ssids**
- int **pending_mic_error_report**
- int **pending_mic_error_pairwise**
- int **mic_errors_seen**
- struct wps_context * **wps**
- int **wps_success**
- struct wps_er * **wps_er**
- int **blacklist_cleared**
- struct wpabuf * **pending_eapol_rx**
- struct os_time **pending_eapol_rx_time**
- u8 **pending_eapol_rx_src** [ETH_ALEN]
- struct ibss_rsn * **ibss_rsn**
- struct wpa_ssid * **bgscan_ssid**
- struct bgscan_ops * **bgscan**
- void * **bgscan_priv**

14.429.1 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.

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

- wpa_supplicant/wpa_supplicant_i.h

14.430 wpabuf Struct Reference

Data Fields

- size_t **size**
- size_t **used**
- u8 * **ext_data**

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

- src/utils/wpabuf.h

14.431 wpas_dbus_callbacks Struct Reference

Data Fields

- struct ctrl_iface_dbus_new_priv *(* **dbus_ctrl_init**)(struct wpa_global *global)
- void(* **dbus_ctrl_deinit**)(struct ctrl_iface_dbus_new_priv *iface)
- void(* **signal_interface_created**)(struct wpa_supplicant *wpa_s)
- void(* **signal_interface_removed**)(struct wpa_supplicant *wpa_s)
- int(* **register_interface**)(struct wpa_supplicant *wpa_s)
- int(* **unregister_interface**)(struct wpa_supplicant *wpa_s)
- void(* **signal_scan_done**)(struct wpa_supplicant *wpa_s, int success)
- void(* **signal_blob_added**)(struct wpa_supplicant *wpa_s, const char *name)
- void(* **signal_blob_removed**)(struct wpa_supplicant *wpa_s, const char *name)
- void(* **signal_network_selected**)(struct wpa_supplicant *wpa_s, int id)
- void(* **signal_state_changed**)(struct wpa_supplicant *wpa_s, wpa_states new_state, wpa_states old_state)
- int(* **register_network**)(struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- int(* **unregister_network**)(struct wpa_supplicant *wpa_s, int nid)
- void(* **signal_network_enabled_changed**)(struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- int(* **register_bss**)(struct wpa_supplicant *wpa_s, u8 bssid[ETH_ALEN])
- int(* **unregister_bss**)(struct wpa_supplicant *wpa_s, u8 bssid[ETH_ALEN])
- void(* **signal_prop_changed**)(struct wpa_supplicant *wpa_s, enum wpas_dbus_prop property)
- void(* **signal_debug_params_changed**)(struct wpa_global *global)

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

- wpa_supplicant/ctrl_iface_dbus_new.h

14.432 wpas_dbus_method Struct Reference

Data Fields

- const char * **name**
- const char * **iface**
- WPADBusMethodHandler **handler**
- struct wpa_dbus_argument **args** [3]

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

- wpa_supplicant/ctrl_iface_dbus_new.c

14.433 wpas_dbus_property Struct Reference

Data Fields

- const char * **name**
- const char * **iface**
- const char * **type**
- WPADBusPropertyAccessor **getter**
- WPADBusPropertyAccessor **setter**
- enum dbus_prop_access **_access**

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

- wpa_supplicant/ctrl_iface_dbus_new.c

14.434 wpas_dbus_signal Struct Reference

Data Fields

- const char * **name**
- const char * **iface**
- struct wpa_dbus_argument **args** [3]

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

- wpa_supplicant/ctrl_iface_dbus_new.c

14.435 wps_config Struct Reference

WPS configuration for a single registration protocol run.

`#include <wps.h>`

Data Fields

- struct wps_context * wps

  Pointer to long term WPS context.

- int registrar

  Whether this end is a Registrar.

- const u8 * pin

  Enrollee Device Password (NULL for Registrar or PBC).

- size_t pin_len

  Length on pin in octets.

- int pbc

  Whether this is protocol run uses PBC.

- struct wpabuf * assoc_wps_ie
- struct wps_credential * new_ap_settings

  New AP settings (NULL if not used).

- const u8 * peer_addr

14.435.1 Detailed Description

WPS configuration for a single registration protocol run.

14.435.2 Field Documentation

14.435.2.1 struct wpabuf* wps_config::assoc_wps_ie [read]

assoc_wps_ie: (Re)AssocReq WPS IE (in AP; NULL if not AP)

14.435.2.2 struct wps_credential* wps_config::new_ap_settings [read]

New AP settings (NULL if not used). This parameter provides new AP settings when using a wireless stations as a Registrar to configure the AP. NULL means that AP will not be reconfigured, i.e., the station will only learn the current AP settings by using AP PIN.

14.435.2.3 const u8* wps_config::peer_addr

peer_addr: MAC address of the peer in AP; NULL if not AP

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

- src/wps/wps.h

14.436 wps_context Struct Reference

Long term WPS context data.

`#include <wps.h>`

Data Fields

- int ap
  Whether the local end is an access point.
- struct wps_registrar * registrar
  Pointer to WPS registrar data from wps_registrar_init().
- enum wps_state wps_state
  Current WPS state.
- int ap_setup_locked
  Whether AP setup is locked (only used at AP).
- u8 uuid [16]
  Own UUID.
- u8 ssid [32]
  SSID.
- size_t ssid_len
  Length of ssid in octets.
- struct wps_device_data dev
  Own WPS device data.
- struct oob_conf_data oob_conf
  OOB Config data.
- u16 oob_dev_pw_id
  OOB Device password id.
- void * dh_ctx
  Context data for Diffie-Hellman operation.
- struct wpabuf * dh_privkey
  Diffie-Hellman private key.
- struct wpabuf * dh_pubkey
  Diffie-Hellman public key.
- u16 config_methods
  Enabled configuration methods.

- u16 encr_types

  Enabled encryption types (bit field of WPS_ENCR_∗).

- u16 auth_types

  Authentication types (bit field of WPS_AUTH_∗).

- u8 ∗ network_key

  The current Network Key (PSK) or NULL to generate new.

- size_t network_key_len

  Length of network_key in octets.

- u8 ∗ ap_settings

  AP Settings override for M7 (only used at AP).

- size_t ap_settings_len

  Length of ap_settings in octets.

- char ∗ friendly_name

  Friendly Name (required for UPnP).

- char ∗ manufacturer_url

  Manufacturer URL (optional for UPnP).

- char ∗ model_description

  Model Description (recommended for UPnP).

- char ∗ model_url

  Model URL (optional for UPnP).

- char ∗ upc

  Universal Product Code (optional for UPnP).

- int(∗ cred_cb)(void ∗ctx, const struct wps_credential ∗cred)

  Callback to notify that new Credentials were received.

- void(∗ event_cb)(void ∗ctx, enum wps_event event, union wps_event_data ∗data)

  Event callback (state information about progress).

- void ∗ cb_ctx
- struct upnp_wps_device_sm ∗ **wps_upnp**
- struct upnp_pending_message ∗ **upnp_msgs**

14.436.1 Detailed Description

Long term WPS context data. This data is stored at the higher layer Authenticator or Supplicant data structures and it is maintained over multiple registration protocol runs.

14.436.2 Field Documentation

14.436.2.1 u8* wps_context::ap_settings

AP Settings override for M7 (only used at AP). If NULL, AP Settings attributes will be generated based on the current network configuration.

14.436.2.2 void* wps_context::cb_ctx

cb_ctx: Higher layer context data for callbacks

14.436.2.3 u16 wps_context::config_methods

Enabled configuration methods. Bit field of WPS_CONFIG_*

14.436.2.4 int(* wps_context::cred_cb)(void *ctx, const struct wps_credential *cred)

Callback to notify that new Credentials were received.

Parameters:

ctx Higher layer context data (cb_ctx)

cred The received Credential Return: 0 on success, -1 on failure

14.436.2.5 void(* wps_context::event_cb)(void *ctx, enum wps_event event, union wps_event_data *data)

Event callback (state information about progress).

Parameters:

ctx Higher layer context data (cb_ctx)

event Event type

data Event data

14.436.2.6 u8* wps_context::network_key

The current Network Key (PSK) or NULL to generate new. If NULL, Registrar will generate per-device PSK. In addition, AP uses this when acting as an Enrollee to notify Registrar of the current configuration.

14.436.2.7 u8 wps_context::ssid[32]

SSID. This SSID is used by the Registrar to fill in information for Credentials. In addition, AP uses it when acting as an Enrollee to notify Registrar of the current configuration.

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

- src/wps/wps.h

14.437 wps_credential Struct Reference

WPS Credential.

`#include <wps.h>`

Data Fields

- u8 **ssid** [32]
- size_t **ssid_len**
- u16 **auth_type**
- u16 **encr_type**
- u8 **key_idx**
- u8 **key** [64]
- size_t **key_len**
- u8 **mac_addr** [ETH_ALEN]
- const u8 * **cred_attr**
- size_t **cred_attr_len**

14.437.1 Detailed Description

WPS Credential.

Parameters:

ssid SSID

ssid_len Length of SSID

auth_type Authentication Type (WPS_AUTH_OPEN, .. flags)

encr_type Encryption Type (WPS_ENCR_NONE, .. flags)

key_idx Key index

key Key

key_len Key length in octets

mac_addr MAC address of the Credential receiver

cred_attr Unparsed Credential attribute data (used only in cred_cb()); this may be NULL, if not used

cred_attr_len Length of cred_attr in octets

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

- src/wps/wps.h

14.438 wps_data Struct Reference

WPS registration protocol data.

`#include <wps_i.h>`

Public Types

- enum {
  SEND_M1, **RECV_M2**, **SEND_M3**, **RECV_M4**,
  SEND_M5, **RECV_M6**, **SEND_M7**, **RECV_M8**,
  RECEIVED_M2D, **WPS_MSG_DONE**, **RECV_ACK**, **WPS_FINISHED**,
  SEND_WSC_NACK, **RECV_M1**, **SEND_M2**, **RECV_M3**,
  SEND_M4, **RECV_M5**, **SEND_M6**, **RECV_M7**,
  SEND_M8, **RECV_DONE**, **SEND_M2D**, **RECV_M2D_ACK** }

Data Fields

- struct wps_context * wps
  Pointer to long term WPS context.
- int registrar
  Whether this end is a Registrar.
- int er
  Whether the local end is an external registrar.
- enum wps_data:: { ... } **state**
- u8 **uuid_e** [WPS_UUID_LEN]
- u8 **uuid_r** [WPS_UUID_LEN]
- u8 **mac_addr_e** [ETH_ALEN]
- u8 **nonce_e** [WPS_NONCE_LEN]
- u8 **nonce_r** [WPS_NONCE_LEN]
- u8 **psk1** [WPS_PSK_LEN]
- u8 **psk2** [WPS_PSK_LEN]
- u8 **snonce** [2 *WPS_SECRET_NONCE_LEN]
- u8 **peer_hash1** [WPS_HASH_LEN]
- u8 **peer_hash2** [WPS_HASH_LEN]
- struct wpabuf * **dh_privkey**
- struct wpabuf * **dh_pubkey_e**
- struct wpabuf * **dh_pubkey_r**
- u8 **authkey** [WPS_AUTHKEY_LEN]
- u8 **keywrapkey** [WPS_KEYWRAPKEY_LEN]
- u8 **emsk** [WPS_EMSK_LEN]
- struct wpabuf * **last_msg**
- u8 * **dev_password**
- size_t **dev_password_len**
- u16 **dev_pw_id**

- int **pbc**
- u8 request_type

  Request Type attribute from (Re)AssocReq.

- u16 encr_type

  Available encryption types.

- u16 auth_type

  Available authentication types.

- u8 * **new_psk**
- size_t **new_psk_len**
- int **wps_pin_revealed**
- struct wps_credential **cred**
- struct wps_device_data **peer_dev**
- u16 config_error

  Configuration Error value to be used in NACK.

- int **ext_reg**
- struct wps_credential * **new_ap_settings**
- void * **dh_ctx**
- void(* **ap_settings_cb**)(void *ctx, const struct wps_credential *cred)
- void * **ap_settings_cb_ctx**
- struct wps_credential * **use_cred**

14.438.1 Detailed Description

WPS registration protocol data. This data is stored at the EAP-WSC server/peer method and it is kept for a single registration protocol run.

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

- src/wps/wps_i.h

14.439 wps_device_data Struct Reference

WPS Device Data.

`#include <wps.h>`

Data Fields

- u8 **mac_addr** [ETH_ALEN]
- char * **device_name**
- char * **manufacturer**
- char * **model_name**
- char * **model_number**
- char * **serial_number**
- u8 **pri_dev_type** [WPS_DEV_TYPE_LEN]
- u32 **os_version**
- u8 **rf_bands**

14.439.1 Detailed Description

WPS Device Data.

Parameters:

mac_addr Device MAC address

device_name Device Name (0..32 octets encoded in UTF-8)

manufacturer Manufacturer (0..64 octets encoded in UTF-8)

model_name Model Name (0..32 octets encoded in UTF-8)

model_number Model Number (0..32 octets encoded in UTF-8)

serial_number Serial Number (0..32 octets encoded in UTF-8)

pri_dev_type Primary Device Type

os_version OS Version

rf_bands RF bands (WPS_RF_24GHZ, WPS_RF_50GHZ flags)

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

- src/wps/wps.h

14.440 wps_er Struct Reference

Data Fields

- struct wps_context * **wps**
- char **ifname** [17]
- char * **mac_addr_text**
- u8 **mac_addr** [ETH_ALEN]
- char * **ip_addr_text**
- unsigned **ip_addr**
- int **multicast_sd**
- int **ssdp_sd**
- struct wps_er_ap * **ap**
- struct http_server * **http_srv**
- int **http_port**
- unsigned int **next_ap_id**
- unsigned int **event_id**

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

- src/wps/wps_er.h

14.441 wps_er_ap Struct Reference

Data Fields

- struct wps_er_ap * **next**
- struct wps_er * **er**
- struct wps_er_sta * **sta**
- struct in_addr **addr**
- char * **location**
- struct http_client * **http**
- struct wps_data * **wps**
- u8 **uuid** [WPS_UUID_LEN]
- u8 **pri_dev_type** [8]
- u8 **wps_state**
- u8 **mac_addr** [ETH_ALEN]
- char * **friendly_name**
- char * **manufacturer**
- char * **manufacturer_url**
- char * **model_description**
- char * **model_name**
- char * **model_number**
- char * **model_url**
- char * **serial_number**
- char * **udn**
- char * **upc**
- char * **scpd_url**
- char * **control_url**
- char * **event_sub_url**
- int **subscribed**
- unsigned int **id**
- struct wps_credential * **ap_settings**
- void(* **m1_handler**)(struct wps_er_ap *ap, struct wpabuf *m1)

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

- src/wps/wps_er.h

14.442 wps_er_sta Struct Reference

Data Fields

- struct wps_er_sta * **next**
- struct wps_er_ap * **ap**
- u8 **addr** [ETH_ALEN]
- u16 **config_methods**
- u8 **uuid** [WPS_UUID_LEN]
- u8 **pri_dev_type** [8]
- u16 **dev_passwd_id**
- int **m1_received**
- char * **manufacturer**
- char * **model_name**
- char * **model_number**
- char * **serial_number**
- char * **dev_name**
- struct wps_data * **wps**
- struct http_client * **http**

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

- src/wps/wps_er.h

14.443 wps_event_ Struct Reference

Data Fields

- struct wps_event_ * **next**
- struct wps_event_ * **prev**
- struct subscription * **s**
- unsigned **subscriber_sequence**
- int **retry**
- struct subscr_addr * **addr**
- struct wpabuf * **data**
- struct http_client * **http_event**

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

- src/wps/wps_upnp_event.c

14.444 wps_event_data Union Reference

`#include <wps.h>`

Data Structures

- struct wps_event_er_ap
- struct wps_event_er_enrollee
- struct wps_event_fail

  Registration failure information.

- struct wps_event_m2d

  M2D event data.

- struct wps_event_pwd_auth_fail

Data Fields

- struct wps_event_data::wps_event_m2d m2d

  M2D event data.

- struct wps_event_data::wps_event_fail fail

  Registration failure information.

- struct wps_event_data::wps_event_pwd_auth_fail **pwd_auth_fail**
- struct wps_event_data::wps_event_er_ap **ap**
- struct wps_event_data::wps_event_er_enrollee **enrollee**

14.444.1 Detailed Description

union wps_event_data - WPS event data

14.444.2 Field Documentation

14.444.2.1 struct wps_event_data::wps_event_fail wps_event_data::fail

Registration failure information.

Parameters:

msg enum wps_msg_type

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

- src/wps/wps.h

14.445 wps_event_data::wps_event_er_ap Struct Reference

Data Fields

- const u8 * **uuid**
- const u8 * **mac_addr**
- const char * **friendly_name**
- const char * **manufacturer**
- const char * **manufacturer_url**
- const char * **model_description**
- const char * **model_name**
- const char * **model_number**
- const char * **model_url**
- const char * **serial_number**
- const char * **upc**
- const u8 * **pri_dev_type**
- u8 **wps_state**

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

- src/wps/wps.h

14.446 wps_event_data::wps_event_er_enrollee Struct Reference

Data Fields

- const u8 * **uuid**
- const u8 * **mac_addr**
- int **m1_received**
- u16 **config_methods**
- u16 **dev_passwd_id**
- const u8 * **pri_dev_type**
- const char * **dev_name**
- const char * **manufacturer**
- const char * **model_name**
- const char * **model_number**
- const char * **serial_number**

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

- src/wps/wps.h

14.447 wps_event_data::wps_event_fail Struct Reference

Registration failure information.

`#include <wps.h>`

Data Fields

- int **msg**

14.447.1 Detailed Description

Registration failure information.

Parameters:

msg enum wps_msg_type

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

- src/wps/wps.h

14.448 wps_event_data::wps_event_m2d Struct Reference

M2D event data.

`#include <wps.h>`

Data Fields

- u16 **config_methods**
- const u8 * **manufacturer**
- size_t **manufacturer_len**
- const u8 * **model_name**
- size_t **model_name_len**
- const u8 * **model_number**
- size_t **model_number_len**
- const u8 * **serial_number**
- size_t **serial_number_len**
- const u8 * **dev_name**
- size_t **dev_name_len**
- const u8 * **primary_dev_type**
- u16 **config_error**
- u16 **dev_password_id**

14.448.1 Detailed Description

M2D event data.

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

- src/wps/wps.h

14.449 wps_event_data::wps_event_pwd_auth_fail Struct Reference

Data Fields

- int **enrollee**
- int **part**

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

- src/wps/wps.h

14.450 wps_nfc_data Struct Reference

Data Fields

- struct oob_nfc_device_data * **oob_nfc_dev**

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

- src/wps/wps_nfc.c

14.451 wps_parse_attr Struct Reference

Data Fields

- const u8 * **version**
- const u8 * **msg_type**
- const u8 * **enrollee_nonce**
- const u8 * **registrar_nonce**
- const u8 * **uuid_r**
- const u8 * **uuid_e**
- const u8 * **auth_type_flags**
- const u8 * **encr_type_flags**
- const u8 * **conn_type_flags**
- const u8 * **config_methods**
- const u8 * **sel_reg_config_methods**
- const u8 * **primary_dev_type**
- const u8 * **rf_bands**
- const u8 * **assoc_state**
- const u8 * **config_error**
- const u8 * **dev_password_id**
- const u8 * **oob_dev_password**
- const u8 * **os_version**
- const u8 * **wps_state**
- const u8 * **authenticator**
- const u8 * **r_hash1**
- const u8 * **r_hash2**
- const u8 * **e_hash1**
- const u8 * **e_hash2**
- const u8 * **r_snonce1**
- const u8 * **r_snonce2**
- const u8 * **e_snonce1**
- const u8 * **e_snonce2**
- const u8 * **key_wrap_auth**
- const u8 * **auth_type**
- const u8 * **encr_type**
- const u8 * **network_idx**
- const u8 * **network_key_idx**
- const u8 * **mac_addr**
- const u8 * **key_prov_auto**
- const u8 * **dot1x_enabled**
- const u8 * **selected_registrar**
- const u8 * **request_type**
- const u8 * **response_type**
- const u8 * **ap_setup_locked**
- const u8 * **manufacturer**
- size_t **manufacturer_len**
- const u8 * **model_name**
- size_t **model_name_len**
- const u8 * **model_number**
- size_t **model_number_len**

- const u8 * **serial_number**
- size_t **serial_number_len**
- const u8 * **dev_name**
- size_t **dev_name_len**
- const u8 * **public_key**
- size_t **public_key_len**
- const u8 * **encr_settings**
- size_t **encr_settings_len**
- const u8 * **ssid**
- size_t **ssid_len**
- const u8 * **network_key**
- size_t **network_key_len**
- const u8 * **eap_type**
- size_t **eap_type_len**
- const u8 * **eap_identity**
- size_t **eap_identity_len**
- const u8 * **cred** [MAX_CRED_COUNT]
- size_t **cred_len** [MAX_CRED_COUNT]
- size_t **num_cred**

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

- src/wps/wps_i.h

14.452 wps_pbc_session Struct Reference

Data Fields

- struct wps_pbc_session * **next**
- u8 **addr** [ETH_ALEN]
- u8 **uuid_e** [WPS_UUID_LEN]
- struct os_time **timestamp**

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

- src/wps/wps_registrar.c

14.453 wps_registrar Struct Reference

Data Fields

- struct wps_context * **wps**
- int **pbc**
- int **selected_registrar**
- int(* **new_psk_cb**)(void *ctx, const u8 *mac_addr, const u8 *psk, size_t psk_len)
- int(* **set_ie_cb**)(void *ctx, const u8 *beacon_ie, size_t beacon_ie_len, const u8 *probe_resp_ie, size_t probe_resp_ie_len)
- void(* **pin_needed_cb**)(void *ctx, const u8 *uuid_e, const struct wps_device_data *dev)
- void(* **reg_success_cb**)(void *ctx, const u8 *mac_addr, const u8 *uuid_e)
- void(* **set_sel_reg_cb**)(void *ctx, int sel_reg, u16 dev_passwd_id, u16 sel_reg_config_methods)
- void * **cb_ctx**
- struct wps_uuid_pin * **pins**
- struct wps_pbc_session * **pbc_sessions**
- int **skip_cred_build**
- struct wpabuf * **extra_cred**
- int **disable_auto_conf**
- int **sel_reg_dev_password_id_override**
- int **sel_reg_config_methods_override**
- int **static_wep_only**
- struct wps_registrar_device * **devices**
- int **force_pbc_overlap**

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

- src/wps/wps_registrar.c

14.454 wps_registrar_config Struct Reference

WPS Registrar configuration.

`#include <wps.h>`

Data Fields

- int(∗ new_psk_cb)(void ∗ctx, const u8 ∗mac_addr, const u8 ∗psk, size_t psk_len)

  Callback for new PSK.

- int(∗ set_ie_cb)(void ∗ctx, const u8 ∗beacon_ie, size_t beacon_ie_len, const u8 ∗probe_resp_ie, size_t probe_resp_ie_len)

  Callback for WPS IE changes.

- void(∗ pin_needed_cb)(void ∗ctx, const u8 ∗uuid_e, const struct wps_device_data ∗dev)

  Callback for requesting a PIN.

- void(∗ reg_success_cb)(void ∗ctx, const u8 ∗mac_addr, const u8 ∗uuid_e)

  Callback for reporting successful registration.

- void(∗ set_sel_reg_cb)(void ∗ctx, int sel_reg, u16 dev_passwd_id, u16 sel_reg_config_methods)

  Callback for reporting selected registrar changes.

- void ∗ cb_ctx
- int skip_cred_build
- const u8 ∗ extra_cred
- size_t extra_cred_len
- int disable_auto_conf

  Disable auto-configuration on first registration.

- int static_wep_only

  Whether the BSS supports only static WEP.

14.454.1 Detailed Description

WPS Registrar configuration.

14.454.2 Field Documentation

14.454.2.1 void∗ wps_registrar_config::cb_ctx

cb_ctx: Higher layer context data for Registrar callbacks

14.454.2.2 int wps_registrar_config::disable_auto_conf

Disable auto-configuration on first registration. By default, the AP that is started in not configured state will generate a random PSK and move to configured state when the first registration protocol run is completed successfully. This option can be used to disable this functionality and leave it up to an external program to take care of configuration. This requires the extra_cred to be set with a suitable Credential and skip_cred_build being used.

14.454.2.3 const u8* wps_registrar_config::extra_cred

extra_cred: Additional Credential attribute(s)

This optional data (set to NULL to disable) can be used to add Credential attribute(s) for other networks into M8. If skip_cred_build is set, this will also override the automatically generated Credential attribute.

14.454.2.4 size_t wps_registrar_config::extra_cred_len

extra_cred_len: Length of extra_cred in octets

14.454.2.5 int(* wps_registrar_config::new_psk_cb)(void *ctx, const u8 *mac_addr, const u8 *psk, size_t psk_len)

Callback for new PSK.

Parameters:

ctx Higher layer context data (cb_ctx)

mac_addr MAC address of the Enrollee

psk The new PSK

psk_len The length of psk in octets

Returns:

0 on success, -1 on failure

This callback is called when a new per-device PSK is provisioned.

14.454.2.6 void(* wps_registrar_config::pin_needed_cb)(void *ctx, const u8 *uuid_e, const struct wps_device_data *dev)

Callback for requesting a PIN.

Parameters:

ctx Higher layer context data (cb_ctx)

uuid_e UUID-E of the unknown Enrollee

dev Device Data from the unknown Enrollee

This callback is called whenever an unknown Enrollee requests to use PIN method and a matching PIN (Device Password) is not found in Registrar data.

14.454.2.7 void(∗ wps_registrar_config::reg_success_cb)(void ∗ctx, const u8 ∗mac_addr, const u8 ∗uuid_e)

Callback for reporting successful registration.

Parameters:

ctx Higher layer context data (cb_ctx)

mac_addr MAC address of the Enrollee

uuid_e UUID-E of the Enrollee

This callback is called whenever an Enrollee completes registration successfully.

14.454.2.8 int(∗ wps_registrar_config::set_ie_cb)(void ∗ctx, const u8 ∗beacon_ie, size_t beacon_ie_len, const u8 ∗probe_resp_ie, size_t probe_resp_ie_len)

Callback for WPS IE changes.

Parameters:

ctx Higher layer context data (cb_ctx)

beacon_ie WPS IE for Beacon

beacon_ie_len WPS IE length for Beacon

probe_resp_ie WPS IE for Probe Response

probe_resp_ie_len WPS IE length for Probe Response

Returns:

0 on success, -1 on failure

This callback is called whenever the WPS IE in Beacon or Probe Response frames needs to be changed (AP only).

14.454.2.9 void(∗ wps_registrar_config::set_sel_reg_cb)(void ∗ctx, int sel_reg, u16 dev_passwd_id, u16 sel_reg_config_methods)

Callback for reporting selected registrar changes.

Parameters:

ctx Higher layer context data (cb_ctx)

sel_reg Whether the Registrar is selected

dev_passwd_id Device Password ID to indicate with method or specific password the Registrar intends to use

sel_reg_config_methods Bit field of active config methods

This callback is called whenever the Selected Registrar state changes (e.g., a new PIN becomes available or PBC is invoked). This callback is only used by External Registrar implementation; set_ie_cb() is used by AP implementation in similar caes, but it provides the full WPS IE data instead of just the minimal Registrar state information.

14.454.2.10 int wps_registrar_config::skip_cred_build

skip_cred_build: Do not build credential

This option can be used to disable internal code that builds Credential attribute into M8 based on the current network configuration and Enrollee capabilities. The extra_cred data will then be used as the Credential(s).

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

- src/wps/wps.h

14.455 wps_registrar_device Struct Reference

Data Fields

- struct wps_registrar_device * **next**
- struct wps_device_data **dev**
- u8 **uuid** [WPS_UUID_LEN]

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

- src/wps/wps_registrar.c

14.456 wps_ufd_data Struct Reference

Data Fields

- int **ufd_fd**

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

- src/wps/wps_ufd.c

14.457 wps_uuid_pin Struct Reference

Data Fields

- struct wps_uuid_pin * **next**
- u8 **uuid** [WPS_UUID_LEN]
- int **wildcard_uuid**
- u8 * **pin**
- size_t **pin_len**
- int **flags**
- struct os_time **expiration**

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

- src/wps/wps_registrar.c

14.458 x509_algorithm_identifier Struct Reference

Data Fields

- struct asn1_oid **oid**

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

- src/tls/x509v3.h

14.459 x509_certificate Struct Reference

Public Types

- enum { **X509_CERT_V1** = 0, **X509_CERT_V2** = 1, **X509_CERT_V3** = 2 }

Data Fields

- struct x509_certificate * **next**
- enum x509_certificate:: { ... } **version**
- unsigned long **serial_number**
- struct x509_algorithm_identifier **signature**
- struct x509_name **issuer**
- struct x509_name **subject**
- os_time_t **not_before**
- os_time_t **not_after**
- struct x509_algorithm_identifier **public_key_alg**
- u8 * **public_key**
- size_t **public_key_len**
- struct x509_algorithm_identifier **signature_alg**
- u8 * **sign_value**
- size_t **sign_value_len**
- unsigned int **extensions_present**
- int **ca**
- unsigned long **path_len_constraint**
- unsigned long **key_usage**
- const u8 * **cert_start**
- size_t **cert_len**
- const u8 * **tbs_cert_start**
- size_t **tbs_cert_len**

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

- src/tls/x509v3.h

14.460 x509_name Struct Reference

Data Fields

- char * **cn**
- char * **c**
- char * **l**
- char * **st**
- char * **o**
- char * **ou**
- char * **email**
- char * **alt_email**
- char * **dns**
- char * **uri**
- u8 * **ip**
- size_t **ip_len**
- struct asn1_oid **rid**

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

- src/tls/x509v3.h

Chapter 15

File Documentation

15.1 hostapd/accounting.c File Reference

hostapd / RADIUS Accounting `#include "includes.h"`

```
#include "hostapd.h"
#include "radius/radius.h"
#include "radius/radius_client.h"
#include "eloop.h"
#include "accounting.h"
#include "ieee802_1x.h"
#include "driver_i.h"
#include "sta_info.h"
```

Defines

- #define **ACCT_DEFAULT_UPDATE_INTERVAL** 300

Functions

- void accounting_sta_start (struct hostapd_data *hapd, struct sta_info *sta)

  Start STA accounting.

- void accounting_sta_interim (struct hostapd_data *hapd, struct sta_info *sta)

  Send a interim STA accounting report.

- void accounting_sta_stop (struct hostapd_data *hapd, struct sta_info *sta)

  Stop STA accounting.

- int accounting_init (struct hostapd_data *hapd)
- void accounting_deinit (struct hostapd_data *hapd)
- int **accounting_reconfig** (struct hostapd_data *hapd, struct hostapd_config *oldconf)

15.1.1 Detailed Description

hostapd / RADIUS Accounting

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi>

15.1.2 Function Documentation

15.1.2.1 void accounting_deinit (struct hostapd_data * *hapd*)

accounting_deinit: Deinitilize accounting : hostapd BSS data

15.1.2.2 int accounting_init (struct hostapd_data * *hapd*)

accounting_init: Initialize accounting : hostapd BSS data Returns: 0 on success, -1 on failure

15.1.2.3 void accounting_sta_interim (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Send a interim STA accounting report.

Parameters:

hapd hostapd BSS data

sta The station

15.1.2.4 void accounting_sta_start (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Start STA accounting.

Parameters:

hapd hostapd BSS data

sta The station

15.1.2.5 void accounting_sta_stop (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Stop STA accounting.

Parameters:

hapd hostapd BSS data

sta The station

15.2 hostapd/accounting.h File Reference

hostapd / RADIUS Accounting

Functions

- void accounting_sta_interim (struct hostapd_data *hapd, struct sta_info *sta)
  Send a interim STA accounting report.
- void accounting_sta_start (struct hostapd_data *hapd, struct sta_info *sta)
  Start STA accounting.
- void accounting_sta_stop (struct hostapd_data *hapd, struct sta_info *sta)
  Stop STA accounting.
- int accounting_init (struct hostapd_data *hapd)
- void accounting_deinit (struct hostapd_data *hapd)
- int **accounting_reconfig** (struct hostapd_data *hapd, struct hostapd_config *oldconf)

15.2.1 Detailed Description

hostapd / RADIUS Accounting

Copyright

Copyright (c) 2002-2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.2.2 Function Documentation

15.2.2.1 void accounting_deinit (struct hostapd_data * *hapd*)

accounting_deinit: Deinitilize accounting : hostapd BSS data

15.2.2.2 int accounting_init (struct hostapd_data * *hapd*)

accounting_init: Initialize accounting : hostapd BSS data Returns: 0 on success, -1 on failure

15.2.2.3 void accounting_sta_interim (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Send a interim STA accounting report.

Parameters:

hapd hostapd BSS data
sta The station

15.2.2.4 void accounting_sta_start (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Start STA accounting.

Parameters:

hapd hostapd BSS data

sta The station

15.2.2.5 void accounting_sta_stop (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Stop STA accounting.

Parameters:

hapd hostapd BSS data

sta The station

15.3 hostapd/ap_list.c File Reference

hostapd / AP table `#include "includes.h"`

```
#include "hostapd.h"
#include "config.h"
#include "ieee802_11.h"
#include "eloop.h"
#include "sta_info.h"
#include "ap_list.h"
#include "hw_features.h"
#include "beacon.h"
#include "drivers/driver.h"
```

Data Structures

- struct ieee80211_frame_info

Enumerations

- enum **ieee80211_phytype** {

  ieee80211_phytype_fhss_dot11_97 = 1, **ieee80211_phytype_dsss_dot11_97** = 2, **ieee80211_phytype_irbaseband** = 3, **ieee80211_phytype_dsss_dot11_b** = 4,

  ieee80211_phytype_pbcc_dot11_b = 5, **ieee80211_phytype_ofdm_dot11_g** = 6, **ieee80211_phytype_pbcc_dot11_g** = 7, **ieee80211_phytype_ofdm_dot11_a** = 8,

  ieee80211_phytype_dsss_dot11_turbog = 255, **ieee80211_phytype_dsss_dot11_turbo** = 256 }

Functions

- struct ap_info * **ap_get_ap** (struct hostapd_iface *iface, u8 *ap)
- int **ap_ap_for_each** (struct hostapd_iface *iface, int(*func)(struct ap_info *s, void *data), void *data)
- void **ap_list_process_beacon** (struct hostapd_iface *iface, struct ieee80211_mgmt *mgmt, struct ieee802_11_elems *elems, struct hostapd_frame_info *fi)
- int **ap_list_init** (struct hostapd_iface *iface)
- void **ap_list_deinit** (struct hostapd_iface *iface)
- int **ap_list_reconfig** (struct hostapd_iface *iface, struct hostapd_config *oldconf)

Variables

- struct ieee80211_frame_info **packed**

15.3.1 Detailed Description

hostapd / AP table

Copyright

Copyright (c) 2002-2003, Jouni Malinen <j@w1.fi> Copyright (c) 2003-2004, Instant802 Networks, Inc. Copyright (c) 2006, Devicescape Software, Inc. Copyright (c) 2007-2008, Intel Corporation

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

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

See README and COPYING for more details.

15.4 hostapd/ap_list.h File Reference

hostapd / AP table

Data Structures

- struct ap_info

Functions

- struct ap_info * **ap_get_ap** (struct hostapd_iface *iface, u8 *sta)
- int **ap_ap_for_each** (struct hostapd_iface *iface, int(*func)(struct ap_info *s, void *data), void *data)
- void **ap_list_process_beacon** (struct hostapd_iface *iface, struct ieee80211_mgmt *mgmt, struct ieee802_11_elems *elems, struct hostapd_frame_info *fi)
- int **ap_list_reconfig** (struct hostapd_iface *iface, struct hostapd_config *oldconf)

15.4.1 Detailed Description

hostapd / AP table

Copyright

Copyright (c) 2002-2003, Jouni Malinen <j@w1.fi> Copyright (c) 2003-2004, Instant802 Networks, Inc. Copyright (c) 2006, Devicescape Software, Inc. Copyright (c) 2007-2008, Intel Corporation

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

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

See README and COPYING for more details.

15.5 hostapd/beacon.c File Reference

hostapd / IEEE 802.11 Management: Beacon and Probe Request/Response `#include "includes.h"`

```
#include "hostapd.h"
#include "ieee802_11.h"
#include "wpa.h"
#include "wme.h"
#include "beacon.h"
#include "hw_features.h"
#include "driver_i.h"
#include "sta_info.h"
#include "wps_hostapd.h"
```

Defines

- #define **MAX_PROBERESP_LEN** 768
- #define **BEACON_HEAD_BUF_SIZE** 256
- #define **BEACON_TAIL_BUF_SIZE** 512

Functions

- void **handle_probe_req** (struct hostapd_data *hapd, struct ieee80211_mgmt *mgmt, size_t len)
- void **ieee802_11_set_beacon** (struct hostapd_data *hapd)
- void **ieee802_11_set_beacons** (struct hostapd_iface *iface)

15.5.1 Detailed Description

hostapd / IEEE 802.11 Management: Beacon and Probe Request/Response

Copyright

Copyright (c) 2002-2004, Instant802 Networks, Inc. Copyright (c) 2005-2006, Devicescape Software, Inc. Copyright (c) 2008, Jouni Malinen <j@w1.fi> Copyright (c) 2007-2008, Intel Corporation

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

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

See README and COPYING for more details.

15.6 hostapd/beacon.h File Reference

hostapd / IEEE 802.11 Management: Beacon and Probe Request/Response

Functions

- void **handle_probe_req** (struct hostapd_data *hapd, struct ieee80211_mgmt *mgmt, size_t len)

15.6.1 Detailed Description

hostapd / IEEE 802.11 Management: Beacon and Probe Request/Response

Copyright

15.7 hostapd/config.c File Reference

hostapd / Configuration file `#include "includes.h"`

```
#include <grp.h>
#include "hostapd.h"
#include "drivers/driver.h"
#include "sha1.h"
#include "eap_server/eap.h"
#include "radius/radius_client.h"
#include "wpa_common.h"
#include "wpa.h"
#include "uuid.h"
#include "eap_common/eap_wsc_common.h"
#include "sta_info.h"
#include "config.h"
```

Defines

- #define **MAX_STA_COUNT** 2007

Enumerations

- enum {

  IEEE80211_TX_QUEUE_DATA0 = 0, **IEEE80211_TX_QUEUE_DATA1** = 1, **IEEE80211_TX_QUEUE_DATA2** = 2, **IEEE80211_TX_QUEUE_DATA3** = 3,

  IEEE80211_TX_QUEUE_DATA4 = 4, **IEEE80211_TX_QUEUE_AFTER_BEACON** = 6, **IEEE80211_TX_QUEUE_BEACON** = 7 }

Functions

- struct hostapd_config * **hostapd_config_defaults** (void)
- int **hostapd_mac_comp** (const void *a, const void *b)
- int **hostapd_mac_comp_empty** (const void *a)
- int **hostapd_setup_wpa_psk** (struct hostapd_bss_config *conf)
- struct hostapd_config * hostapd_config_read (const char *fname)

  Read and parse a configuration file.

- int **hostapd_wep_key_cmp** (struct hostapd_wep_keys *a, struct hostapd_wep_keys *b)
- void hostapd_config_free (struct hostapd_config *conf)

  Free hostapd configuration.

- int hostapd_maclist_found (struct mac_acl_entry *list, int num_entries, const u8 *addr, int *vlan_id)

  Find a MAC address from a list.

- int **hostapd_rate_found** (int *list, int rate)
- const char * **hostapd_get_vlan_id_ifname** (struct hostapd_vlan *vlan, int vlan_id)
- const u8 * **hostapd_get_psk** (const struct hostapd_bss_config *conf, const u8 *addr, const u8 *prev_psk)
- struct hostapd_eap_user * **hostapd_get_eap_user** (const struct hostapd_bss_config *conf, const u8 *identity, size_t identity_len, int phase2)

Variables

- struct wpa_driver_ops * **wpa_drivers** [ ]

15.7.1 Detailed Description

hostapd / Configuration file

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi> Copyright (c) 2007-2008, Intel Corporation

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

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

See README and COPYING for more details.

15.7.2 Function Documentation

15.7.2.1 void hostapd_config_free (struct hostapd_config * *conf*)

Free hostapd configuration.

Parameters:

conf Configuration data from hostapd_config_read().

15.7.2.2 struct hostapd_config* hostapd_config_read (const char * *fname*) [read]

Read and parse a configuration file.

Parameters:

fname Configuration file name (including path, if needed)

Returns:

Allocated configuration data structure

15.7.2.3 int hostapd_maclist_found (struct mac_acl_entry * *list*, int *num_entries*, const u8 * *addr*, int * *vlan_id*)

Find a MAC address from a list.

Parameters:

list MAC address list

num_entries Number of addresses in the list

addr Address to search for

vlan_id Buffer for returning VLAN ID or NULL if not needed

Returns:

1 if address is in the list or 0 if not.

Perform a binary search for given MAC address from a pre-sorted list.

15.8 wpa_supplicant/config.c File Reference

WPA Supplicant / Configuration parser and common functions. `#include "includes.h"`

```
#include "common.h"
#include "wpa.h"
#include "sha1.h"
#include "eap_peer/eap.h"
#include "config.h"
```

Data Structures

- struct parse_data

Defines

- #define **OFFSET**(v) ((void *) &((struct wpa_ssid *) 0)->v)
- #define **_STR**(f) #f, wpa_config_parse_str, wpa_config_write_str, OFFSET(f)
- #define **_STRe**(f) #f, wpa_config_parse_str, wpa_config_write_str, OFFSET(eap.f)
- #define **STR**(f) _STR(f), NULL, NULL, NULL, 0
- #define **STRe**(f) _STRe(f), NULL, NULL, NULL, 0
- #define **STR_KEY**(f) _STR(f), NULL, NULL, NULL, 1
- #define **STR_KEYe**(f) _STRe(f), NULL, NULL, NULL, 1
- #define **_STR_LEN**(f) _STR(f), OFFSET(f ## _len)
- #define **_STR_LENe**(f) _STRe(f), OFFSET(eap.f ## _len)
- #define **STR_LEN**(f) _STR_LEN(f), NULL, NULL, 0
- #define **STR_LENe**(f) _STR_LENe(f), NULL, NULL, 0
- #define **STR_LEN_KEY**(f) _STR_LEN(f), NULL, NULL, 1
- #define **_STR_RANGE**(f, min, max) _STR_LEN(f), (void *) (min), (void *) (max)
- #define **STR_RANGE**(f, min, max) _STR_RANGE(f, min, max), 0
- #define **STR_RANGE_KEY**(f, min, max) _STR_RANGE(f, min, max), 1
- #define **_INT**(f)
- #define **_INTe**(f)
- #define **INT**(f) _INT(f), NULL, NULL, 0
- #define **INTe**(f) _INTe(f), NULL, NULL, 0
- #define **INT_RANGE**(f, min, max) _INT(f), (void *) (min), (void *) (max), 0
- #define **_FUNC**(f)
- #define **FUNC**(f) _FUNC(f), 0
- #define **FUNC_KEY**(f) _FUNC(f), 1
- #define **NUM_SSID_FIELDS** (sizeof(ssid_fields) / sizeof(ssid_fields[0]))

Functions

- int wpa_config_add_prio_network (struct wpa_config *config, struct wpa_ssid *ssid)

  Add a network to priority lists.

- void wpa_config_free_ssid (struct wpa_ssid *ssid)

  Free network/ssid configuration data.

- void wpa_config_free (struct wpa_config *config)

  Free configuration data.

- struct wpa_ssid * wpa_config_get_network (struct wpa_config *config, int id)

  Get configured network based on id.

- struct 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_all (struct wpa_ssid *ssid, int get_keys)

  Get all options from network configuration.

- char * wpa_config_get (struct wpa_ssid *ssid, const char *var)

  Get a variable in network configuration.

- char * wpa_config_get_no_key (struct wpa_ssid *ssid, const char *var)

  Get a variable in network configuration (no keys).

- void wpa_config_update_psk (struct wpa_ssid *ssid)

  Update WPA PSK based on passphrase and SSID.

- 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.

- struct wpa_config * wpa_config_alloc_empty (const char *ctrl_interface, const char *driver_-param)

  Allocate an empty configuration.

- void wpa_config_debug_dump_networks (struct wpa_config *config)

  Debug dump of configured networks.

15.8.1 Detailed Description

WPA Supplicant / Configuration parser and common functions.

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.8.2 Define Documentation

15.8.2.1 #define _FUNC(f)

Value:

```
#f, wpa_config_parse_ ## f, wpa_config_write_ ## f, \
        NULL, NULL, NULL, NULL
```

15.8.2.2 #define _INT(f)

Value:

```
#f, wpa_config_parse_int, wpa_config_write_int, \
        OFFSET(f), (void *) 0
```

15.8.2.3 #define _INTe(f)

Value:

```
#f, wpa_config_parse_int, wpa_config_write_int, \
        OFFSET(eap.f), (void *) 0
```

15.8.3 Function Documentation

15.8.3.1 struct wpa_ssid* wpa_config_add_network (struct wpa_config * *config*) [read]

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

15.8.3.2 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().

15.8.3.3 struct wpa_config* wpa_config_alloc_empty (const char * *ctrl_interface*, const char * *driver_param*) [read]

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

15.8.3.4 void wpa_config_debug_dump_networks (struct wpa_config * *config*)

Debug dump of configured networks.

Parameters:

config Configuration data from wpa_config_read()

15.8.3.5 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().

15.8.3.6 void wpa_config_free_blob (struct wpa_config_blob * *blob*)

Free blob data.

Parameters:

blob Pointer to blob to be freed

15.8.3.7 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.

15.8.3.8 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.

15.8.3.9 char** wpa_config_get_all (struct wpa_ssid * *ssid*, int *get_keys*)

Get all options from network configuration.

Parameters:

ssid Pointer to network configuration data

get_keys Determines if keys/passwords will be included in returned list

Returns:

NULL terminated list of all set keys and their values in the form of [key1, val1, key2, val2, ... , NULL]

This function can be used to get list of all configured network properties. The caller is responsible for freeing the returned list and all its elements.

15.8.3.10 struct wpa_config_blob* wpa_config_get_blob (struct wpa_config * *config*, const char * *name*) [read]

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

15.8.3.11 struct wpa_ssid* wpa_config_get_network (struct wpa_config * *config*, int *id*) [read]

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

15.8.3.12 char* wpa_config_get_no_key (struct wpa_ssid * *ssid*, const char * *var*)

Get a variable in network configuration (no keys).

Parameters:

ssid Pointer to network configuration data

var Variable name, e.g., "ssid"

Returns:

Value of the variable or NULL on failure

This function can be used to get network configuration variable like wpa_config_get(). The only difference is that this functions does not expose key/password material from the configuration. In case a key/password field is requested, the returned value is an empty string or NULL if the variable is not set or "*" if the variable is set (regardless of its value). The returned value is a copy of the configuration variable in text format, i.e,. the same format that the text-based configuration file and wpa_config_set() are using for the value. The caller is responsible for freeing the returned value.

15.8.3.13 int wpa_config_remove_blob (struct wpa_config * *config*, const char * *name*)

Remove a named configuration blob.

Parameters:

config Configuration data from wpa_config_read()

name Name of the blob to remove

Returns:

0 if blob was removed or -1 if blob was not found

15.8.3.14 int wpa_config_remove_network (struct wpa_config * *config*, int *id*)

Remove a configured network based on id.

Parameters:

config Configuration data from wpa_config_read()

id Unique network id to search for

Returns:

0 on success, or -1 if the network was not found

15.8.3.15 int wpa_config_set (struct wpa_ssid * *ssid*, const char * *var*, const char * *value*, int *line*)

Set a variable in network configuration.

Parameters:

ssid Pointer to network configuration data

var Variable name, e.g., "ssid"

value Variable value

line Line number in configuration file or 0 if not used

Returns:

0 on success, -1 on failure

This function can be used to set network configuration variables based on both the configuration file and management interface input. The value parameter must be in the same format as the text-based configuration file is using. For example, strings are using double quotation marks.

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

Set or add a named configuration blob.

Parameters:

config Configuration data from wpa_config_read()

blob New value for the blob

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

15.8.3.17 void wpa_config_set_network_defaults (struct wpa_ssid * *ssid*)

Set network default values.

Parameters:

ssid Pointer to network configuration data

15.8.3.18 void wpa_config_update_psk (struct wpa_ssid * *ssid*)

Update WPA PSK based on passphrase and SSID.

Parameters:

ssid Pointer to network configuration data

This function must be called to update WPA PSK when either SSID or the passphrase has changed for the network configuration.

15.9 hostapd/config.h File Reference

hostapd / Configuration file `#include "defs.h"`

```
#include "ip_addr.h"
#include "wpa_common.h"
```

Data Structures

- struct mac_acl_entry
- struct hostapd_wep_keys
- struct hostapd_ssid
- struct hostapd_vlan
- struct hostapd_wpa_psk
- struct hostapd_eap_user
- struct hostapd_tx_queue_params
- struct hostapd_wmm_ac_params
- struct hostapd_bss_config

  Per-BSS configuration.

- struct hostapd_config

  Per-radio interface configuration.

Defines

- #define **HOSTAPD_MAX_SSID_LEN** 32
- #define **NUM_WEP_KEYS** 4
- #define **DYNAMIC_VLAN_DISABLED** 0
- #define **DYNAMIC_VLAN_OPTIONAL** 1
- #define **DYNAMIC_VLAN_REQUIRED** 2
- #define **VLAN_ID_WILDCARD** -1
- #define **PMK_LEN** 32
- #define **EAP_USER_MAX_METHODS** 8
- #define **NUM_TX_QUEUES** 8

Typedefs

- typedef u8 **macaddr** [ETH_ALEN]
- typedef enum hostap_security_policy **secpolicy**

Enumerations

- enum **hostap_security_policy** {

  SECURITY_PLAINTEXT = 0, **SECURITY_STATIC_WEP** = 1, **SECURITY_IEEE_802_1X** = 2, **SECURITY_WPA_PSK** = 3,

  SECURITY_WPA = 4 }

Functions

- int **hostapd_mac_comp** (const void *a, const void *b)
- int **hostapd_mac_comp_empty** (const void *a)
- struct hostapd_config * **hostapd_config_defaults** (void)
- struct hostapd_config * hostapd_config_read (const char *fname)

  Read and parse a configuration file.

- void hostapd_config_free (struct hostapd_config *conf)

  Free hostapd configuration.

- int hostapd_maclist_found (struct mac_acl_entry *list, int num_entries, const u8 *addr, int *vlan_id)

  Find a MAC address from a list.

- int **hostapd_rate_found** (int *list, int rate)
- int **hostapd_wep_key_cmp** (struct hostapd_wep_keys *a, struct hostapd_wep_keys *b)
- const u8 * **hostapd_get_psk** (const struct hostapd_bss_config *conf, const u8 *addr, const u8 *prev_psk)
- int **hostapd_setup_wpa_psk** (struct hostapd_bss_config *conf)
- const char * **hostapd_get_vlan_id_ifname** (struct hostapd_vlan *vlan, int vlan_id)
- struct hostapd_eap_user * **hostapd_get_eap_user** (const struct hostapd_bss_config *conf, const u8 *identity, size_t identity_len, int phase2)

15.9.1 Detailed Description

hostapd / Configuration file

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi> Copyright (c) 2007-2008, Intel Corporation

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

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

See README and COPYING for more details.

15.9.2 Function Documentation

15.9.2.1 void hostapd_config_free (struct hostapd_config * *conf*)

Free hostapd configuration.

Parameters:

conf Configuration data from hostapd_config_read().

15.9.2.2 struct hostapd_config* hostapd_config_read (const char * *fname*) [read]

Read and parse a configuration file.

Parameters:

fname Configuration file name (including path, if needed)

Returns:

Allocated configuration data structure

15.9.2.3 int hostapd_maclist_found (struct mac_acl_entry * *list*, int *num_entries*, const u8 * *addr*, int * *vlan_id*)

Find a MAC address from a list.

Parameters:

list MAC address list

num_entries Number of addresses in the list

addr Address to search for

vlan_id Buffer for returning VLAN ID or NULL if not needed

Returns:

1 if address is in the list or 0 if not.

Perform a binary search for given MAC address from a pre-sorted list.

15.10 wpa_supplicant/config.h File Reference

WPA Supplicant / Configuration file structures. `#include "config_ssid.h"`

Data Structures

- struct wpa_config

  wpa_supplicant configuration data

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.

- struct wpa_ssid * wpa_config_get_network (struct wpa_config *config, int id)

  Get configured network based on id.

- struct 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_all (struct wpa_ssid *ssid, int get_keys)

  Get all options from network configuration.

- char * wpa_config_get (struct wpa_ssid *ssid, const char *var)

  Get a variable in network configuration.

- char * wpa_config_get_no_key (struct wpa_ssid *ssid, const char *var)

  Get a variable in network configuration (no keys).

- void wpa_config_update_psk (struct wpa_ssid *ssid)
  Update WPA PSK based on passphrase and SSID.
- int wpa_config_add_prio_network (struct wpa_config *config, struct wpa_ssid *ssid)
  Add a network to priority lists.
- 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.
- struct wpa_config * wpa_config_alloc_empty (const char *ctrl_interface, const char *driver_-param)
  Allocate an empty configuration.
- void wpa_config_debug_dump_networks (struct wpa_config *config)
  Debug dump of configured networks.
- struct 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.

15.10.1 Detailed Description

WPA Supplicant / Configuration file structures.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.10.2 Function Documentation

15.10.2.1 struct wpa_ssid* wpa_config_add_network (struct wpa_config * *config*) [read]

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

15.10.2.2 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().

15.10.2.3 struct wpa_config* wpa_config_alloc_empty (const char * *ctrl_interface*, const char * *driver_param*) [read]

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

15.10.2.4 void wpa_config_debug_dump_networks (struct wpa_config * *config*)

Debug dump of configured networks.

Parameters:

config Configuration data from wpa_config_read()

15.10.2.5 void wpa_config_free (struct wpa_config * *config*)

Free configuration data.

Parameters:

config Configuration data from wpa_config_read()

This function frees all resources allocated for the configuration data by wpa_config_read().

15.10.2.6 void wpa_config_free_blob (struct wpa_config_blob * *blob*)

Free blob data.

Parameters:

blob Pointer to blob to be freed

15.10.2.7 void wpa_config_free_ssid (struct wpa_ssid * *ssid*)

Free network/ssid configuration data.

Parameters:

ssid Configuration data for the network

This function frees all resources allocated for the network configuration data.

15.10.2.8 char* wpa_config_get (struct wpa_ssid * *ssid*, const char * *var*)

Get a variable in network configuration.

Parameters:

ssid Pointer to network configuration data

var Variable name, e.g., "ssid"

Returns:

Value of the variable or NULL on failure

This function can be used to get network configuration variables. The returned value is a copy of the configuration variable in text format, i.e,. the same format that the text-based configuration file and wpa_config_set() are using for the value. The caller is responsible for freeing the returned value.

15.10.2.9 char** wpa_config_get_all (struct wpa_ssid * *ssid*, int *get_keys*)

Get all options from network configuration.

Parameters:

ssid Pointer to network configuration data

get_keys Determines if keys/passwords will be included in returned list

Returns:

NULL terminated list of all set keys and their values in the form of [key1, val1, key2, val2, ... , NULL]

This function can be used to get list of all configured network properties. The caller is responsible for freeing the returned list and all its elements.

15.10.2.10 struct wpa_config_blob* wpa_config_get_blob (struct wpa_config * *config*, const char * *name*) [read]

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

15.10.2.11 struct wpa_ssid* wpa_config_get_network (struct wpa_config * *config*, int *id*) [read]

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

15.10.2.12 char* wpa_config_get_no_key (struct wpa_ssid * *ssid*, const char * *var*)

Get a variable in network configuration (no keys).

Parameters:

ssid Pointer to network configuration data

var Variable name, e.g., "ssid"

Returns:

Value of the variable or NULL on failure

This function can be used to get network configuration variable like wpa_config_get(). The only difference is that this functions does not expose key/password material from the configuration. In case a key/password field is requested, the returned value is an empty string or NULL if the variable is not set or "*" if the variable is set (regardless of its value). The returned value is a copy of the configuration variable in text format, i.e,. the same format that the text-based configuration file and wpa_config_set() are using for the value. The caller is responsible for freeing the returned value.

15.10.2.13 struct wpa_config* wpa_config_read (const char * *name*) [read]

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.

15.10.2.14 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

15.10.2.15 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

15.10.2.16 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.

15.10.2.17 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.

15.10.2.18 void wpa_config_set_network_defaults (struct wpa_ssid * *ssid*)

Set network default values.

Parameters:

ssid Pointer to network configuration data

15.10.2.19 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.

15.10.2.20 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.

15.11 hostapd/ctrl_iface.c File Reference

hostapd / UNIX domain socket -based control interface `#include "includes.h"`

```
#include <sys/un.h>
#include <sys/stat.h>
#include <stddef.h>
#include "hostapd.h"
#include "eloop.h"
#include "config.h"
#include "ieee802_1x.h"
#include "wpa.h"
#include "radius/radius_client.h"
#include "ieee802_11.h"
#include "ctrl_iface.h"
#include "sta_info.h"
#include "accounting.h"
#include "wps_hostapd.h"
#include "drivers/driver.h"
#include "ctrl_iface_ap.h"
```

Data Structures

- struct wpa_ctrl_dst

  Internal data structure of control interface clients.

Functions

- int **hostapd_ctrl_iface_init** (struct hostapd_data *hapd)
- void **hostapd_ctrl_iface_deinit** (struct hostapd_data *hapd)

15.11.1 Detailed Description

hostapd / UNIX domain socket -based control interface

Copyright

15.12 wpa_supplicant/ctrl_iface.c File Reference

WPA Supplicant / Control interface (shared code for all backends). `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "wpa.h"
#include "config.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "ctrl_iface.h"
#include "l2_packet/l2_packet.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_ctrl.h"
#include "eap_peer/eap.h"
#include "ieee802_11_defs.h"
#include "wps_supplicant.h"
#include "wps/wps.h"
#include "ibss_rsn.h"
#include "ap.h"
#include "notify.h"
```

Functions

- char * wpa_supplicant_ctrl_iface_process (struct wpa_supplicant *wpa_s, char *buf, size_t *resp_len)

  Process ctrl_iface command.

- char * wpa_supplicant_global_ctrl_iface_process (struct wpa_global *global, char *buf, size_t *resp_len)

  Process global ctrl_iface command.

Variables

- struct wpa_driver_ops * **wpa_drivers** [ ]

15.12.1 Detailed Description

WPA Supplicant / Control interface (shared code for all backends).

Copyright

15.12.2 Function Documentation

15.12.2.1 char* wpa_supplicant_ctrl_iface_process (struct wpa_supplicant * *wpa_s*, char * *buf*, size_t * *resp_len*)

Process ctrl_iface command.

Parameters:

wpa_s Pointer to wpa_supplicant data

buf Received command buffer (nul terminated string)

resp_len Variable to be set to the response length

Returns:

Response (*resp_len bytes) or NULL on failure

Control interface backends call this function when receiving a message that they do not process internally, i.e., anything else than ATTACH, DETACH, and LEVEL. The return response value is then sent to the external program that sent the command. Caller is responsible for freeing the buffer after this. If NULL is returned, *resp_len can be set to two special values: 1 = send "FAIL\n" response, 2 = send "OK\n" response. If *resp_len has any other value, no response is sent.

15.12.2.2 char* wpa_supplicant_global_ctrl_iface_process (struct wpa_global * *global*, char * *buf*, size_t * *resp_len*)

Process global ctrl_iface command.

Parameters:

global Pointer to global data from wpa_supplicant_init()

buf Received command buffer (nul terminated string)

resp_len Variable to be set to the response length

Returns:

Response (*resp_len bytes) or NULL on failure

Control interface backends call this function when receiving a message from the global ctrl_iface connection. The return response value is then sent to the external program that sent the command. Caller is responsible for freeing the buffer after this. If NULL is returned, *resp_len can be set to two special values: 1 = send "FAIL\n" response, 2 = send "OK\n" response. If *resp_len has any other value, no response is sent.

15.13 hostapd/ctrl_iface.h File Reference

hostapd / UNIX domain socket -based control interface

Functions

- int **hostapd_ctrl_iface_init** (struct hostapd_data *hapd)
- void **hostapd_ctrl_iface_deinit** (struct hostapd_data *hapd)

15.13.1 Detailed Description

hostapd / UNIX domain socket -based control interface

Copyright

15.14 wpa_supplicant/ctrl_iface.h File Reference

WPA Supplicant / UNIX domain socket -based control interface.

Functions

- char * wpa_supplicant_ctrl_iface_process (struct wpa_supplicant *wpa_s, char *buf, size_t *resp_len)

  Process ctrl_iface command.

- char * wpa_supplicant_global_ctrl_iface_process (struct wpa_global *global, char *buf, size_t *resp_len)

  Process global ctrl_iface command.

- struct ctrl_iface_priv * wpa_supplicant_ctrl_iface_init (struct wpa_supplicant *wpa_s)

  Initialize control interface.

- void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv *priv)

  Deinitialize control interface.

- void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv *priv)

  Wait for ctrl_iface monitor.

- struct ctrl_iface_global_priv * wpa_supplicant_global_ctrl_iface_init (struct wpa_global *global)

  Initialize global control interface.

- void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv *priv)

  Deinitialize global ctrl interface.

15.14.1 Detailed Description

WPA Supplicant / UNIX domain socket -based control interface.

Copyright

Copyright (c) 2004-2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.14.2 Function Documentation

15.14.2.1 void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv * *priv*)

Deinitialize control interface.

Parameters:

priv Pointer to private data from wpa_supplicant_ctrl_iface_init()

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

Required to be implemented in each control interface backend.

15.14.2.2 struct ctrl_iface_priv* wpa_supplicant_ctrl_iface_init (struct wpa_supplicant * *wpa_s*) [read]

Initialize control interface.

Parameters:

wpa_s Pointer to wpa_supplicant data

Returns:

Pointer to private data on success, NULL on failure

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

Required to be implemented in each control interface backend.

15.14.2.3 char* wpa_supplicant_ctrl_iface_process (struct wpa_supplicant * *wpa_s*, char * *buf*, size_t * *resp_len*)

Process ctrl_iface command.

Parameters:

wpa_s Pointer to wpa_supplicant data

buf Received command buffer (nul terminated string)

resp_len Variable to be set to the response length

Returns:

Response (*resp_len bytes) or NULL on failure

Control interface backends call this function when receiving a message that they do not process internally, i.e., anything else than ATTACH, DETACH, and LEVEL. The return response value is then sent to the external program that sent the command. Caller is responsible for freeing the buffer after this. If NULL is returned, *resp_len can be set to two special values: 1 = send "FAIL\n" response, 2 = send "OK\n" response. If *resp_len has any other value, no response is sent.

15.14.2.4 void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv * *priv*)

Wait for ctrl_iface monitor.

Parameters:

priv Pointer to private data from wpa_supplicant_ctrl_iface_init()

Wait until the first message from an external program using the control interface is received. This function can be used to delay normal startup processing to allow control interface programs to attach with wpa_supplicant before normal operations are started.

Required to be implemented in each control interface backend.

15.14.2.5 void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv * *priv*)

Deinitialize global ctrl interface.

Parameters:

priv Pointer to private data from wpa_supplicant_global_ctrl_iface_init()

Deinitialize the global control interface that was initialized with wpa_supplicant_global_ctrl_iface_init().

Required to be implemented in each control interface backend.

15.14.2.6 struct ctrl_iface_global_priv* wpa_supplicant_global_ctrl_iface_init (struct wpa_global * *global*) [read]

Initialize global control interface.

Parameters:

global Pointer to global data from wpa_supplicant_init()

Returns:

Pointer to private data on success, NULL on failure

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

Required to be implemented in each control interface backend.

15.14.2.7 char* wpa_supplicant_global_ctrl_iface_process (struct wpa_global * *global*, char * *buf*, size_t * *resp_len*)

Process global ctrl_iface command.

Parameters:

global Pointer to global data from wpa_supplicant_init()

buf Received command buffer (nul terminated string)

resp_len Variable to be set to the response length

Returns:

Response (*resp_len bytes) or NULL on failure

Control interface backends call this function when receiving a message from the global ctrl_iface connection. The return response value is then sent to the external program that sent the command. Caller is responsible for freeing the buffer after this. If NULL is returned, *resp_len can be set to two special values: 1 = send "FAIL\n" response, 2 = send "OK\n" response. If *resp_len has any other value, no response is sent.

15.15 hostapd/ctrl_iface_ap.c File Reference

Control interface for shared AP commands. `#include "includes.h"`

```
#include "hostapd.h"
#include "ieee802_1x.h"
#include "wpa.h"
#include "ieee802_11.h"
#include "sta_info.h"
#include "wps_hostapd.h"
#include "ctrl_iface_ap.h"
```

Functions

- int **hostapd_ctrl_iface_sta_first** (struct hostapd_data *hapd, char *buf, size_t buflen)
- int **hostapd_ctrl_iface_sta** (struct hostapd_data *hapd, const char *txtaddr, char *buf, size_t buflen)
- int **hostapd_ctrl_iface_sta_next** (struct hostapd_data *hapd, const char *txtaddr, char *buf, size_t buflen)

15.15.1 Detailed Description

Control interface for shared AP commands.

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.16 hostapd/ctrl_iface_ap.h File Reference

Control interface for shared AP commands.

Functions

- int **hostapd_ctrl_iface_sta_first** (struct hostapd_data *hapd, char *buf, size_t buflen)
- int **hostapd_ctrl_iface_sta** (struct hostapd_data *hapd, const char *txtaddr, char *buf, size_t buflen)
- int **hostapd_ctrl_iface_sta_next** (struct hostapd_data *hapd, const char *txtaddr, char *buf, size_t buflen)

15.16.1 Detailed Description

Control interface for shared AP commands.

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.17 hostapd/driver_i.h File Reference

hostapd - internal driver interface wrappers `#include "drivers/driver.h"`

`#include "config.h"`

15.17.1 Detailed Description

hostapd - internal driver interface wrappers

Copyright

15.18 wpa_supplicant/driver_i.h File Reference

wpa_supplicant - Internal driver interface wrappers `#include "drivers/driver.h"`

15.18.1 Detailed Description

wpa_supplicant - Internal driver interface wrappers

Copyright

15.19 hostapd/drv_callbacks.c File Reference

hostapd / Callback functions for driver wrappers `#include "includes.h"`

```
#include "hostapd.h"
#include "driver_i.h"
#include "ieee802_11.h"
#include "radius/radius.h"
#include "sta_info.h"
#include "accounting.h"
#include "tkip_countermeasures.h"
#include "ieee802_1x.h"
#include "wpa.h"
#include "iapp.h"
#include "wme.h"
#include "wps_hostapd.h"
```

Data Structures

- struct prune_data

Defines

- #define **HAPD_BROADCAST** ((struct hostapd_data *) -1)

Functions

- void hostapd_new_assoc_sta (struct hostapd_data *hapd, struct sta_info *sta, int reassoc)

  Notify that a new station associated with the AP.

- void **hostapd_tx_status** (struct hostapd_data *hapd, const u8 *addr, const u8 *buf, size_t len, int ack)
- void **hostapd_rx_from_unknown_sta** (struct hostapd_data *hapd, const struct ieee80211_hdr *hdr, size_t len)
- int **hostapd_notif_assoc** (struct hostapd_data *hapd, const u8 *addr, const u8 *ie, size_t ielen)
- void **hostapd_notif_disassoc** (struct hostapd_data *hapd, const u8 *addr)
- void **hostapd_eapol_receive** (struct hostapd_data *hapd, const u8 *sa, const u8 *buf, size_t len)
- void **hostapd_michael_mic_failure** (struct hostapd_data *hapd, const u8 *addr)
- struct hostapd_data * **hostapd_sta_get_bss** (struct hostapd_data *hapd, const u8 *addr)
- void wpa_supplicant_event (void *ctx, wpa_event_type event, union wpa_event_data *data)

  Report a driver event for wpa_supplicant.

- void **hostapd_probe_req_rx** (struct hostapd_data *hapd, const u8 *sa, const u8 *ie, size_t ie_len)

15.19.1 Detailed Description

hostapd / Callback functions for driver wrappers

Copyright

Copyright (c) 2002-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.19.2 Function Documentation

15.19.2.1 void hostapd_new_assoc_sta (struct hostapd_data * *hapd*, struct sta_info * *sta*, int *reassoc*)

Notify that a new station associated with the AP.

Parameters:

hapd Pointer to BSS data

sta Pointer to the associated STA data

reassoc 1 to indicate this was a re-association; 0 = first association

This function will be called whenever a station associates with the AP. It can be called from ieee802_11.c for drivers that export MLME to hostapd and from driver_*.c for drivers that take care of management frames (IEEE 802.11 authentication and association) internally.

15.19.2.2 void wpa_supplicant_event (void * *ctx*, wpa_event_type *event*, union wpa_event_data * *data*)

Report a driver event for wpa_supplicant.

Parameters:

ctx Context pointer (wpa_s); 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.

15.20 hostapd/eapol_sm.c File Reference

hostapd / IEEE 802.1X-2004 Authenticator - EAPOL state machine `#include "includes.h"`

```
#include "common.h"
#include "ieee802_1x.h"
#include "eapol_sm.h"
#include "eloop.h"
#include "wpa.h"
#include "preauth.h"
#include "sta_info.h"
#include "eap_server/eap.h"
#include "state_machine.h"
#include "eap_common/eap_common.h"
```

Defines

- #define **STATE_MACHINE_DATA** struct eapol_state_machine
- #define **STATE_MACHINE_DEBUG_PREFIX** "IEEE 802.1X"
- #define **STATE_MACHINE_ADDR** sm->addr
- #define **setPortAuthorized**() sm->eapol->cb.set_port_authorized(sm->hapd, sm->sta, 1)
- #define **setPortUnauthorized**() sm->eapol->cb.set_port_authorized(sm->hapd, sm->sta, 0)
- #define **txCannedFail**() eapol_auth_tx_canned_eap(sm, 0)
- #define **txCannedSuccess**() eapol_auth_tx_canned_eap(sm, 1)
- #define **txReq**() eapol_auth_tx_req(sm)
- #define **abortAuth**() sm->eapol->cb.abort_auth(sm->hapd, sm->sta)
- #define **txKey**() sm->eapol->cb.tx_key(sm->hapd, sm->sta)
- #define **processKey**() do { } while (0)

Functions

- **SM_STATE** (AUTH_PAE, INITIALIZE)
- **SM_STATE** (AUTH_PAE, DISCONNECTED)
- **SM_STATE** (AUTH_PAE, RESTART)
- **SM_STATE** (AUTH_PAE, CONNECTING)
- **SM_STATE** (AUTH_PAE, HELD)
- **SM_STATE** (AUTH_PAE, AUTHENTICATED)
- **SM_STATE** (AUTH_PAE, AUTHENTICATING)
- **SM_STATE** (AUTH_PAE, ABORTING)
- **SM_STATE** (AUTH_PAE, FORCE_AUTH)
- **SM_STATE** (AUTH_PAE, FORCE_UNAUTH)
- **SM_STEP** (AUTH_PAE)
- **SM_STATE** (BE_AUTH, INITIALIZE)
- **SM_STATE** (BE_AUTH, REQUEST)
- **SM_STATE** (BE_AUTH, RESPONSE)
- **SM_STATE** (BE_AUTH, SUCCESS)

- **SM_STATE** (BE_AUTH, FAIL)
- **SM_STATE** (BE_AUTH, TIMEOUT)
- **SM_STATE** (BE_AUTH, IDLE)
- **SM_STATE** (BE_AUTH, IGNORE)
- **SM_STEP** (BE_AUTH)
- **SM_STATE** (REAUTH_TIMER, INITIALIZE)
- **SM_STATE** (REAUTH_TIMER, REAUTHENTICATE)
- **SM_STEP** (REAUTH_TIMER)
- **SM_STATE** (AUTH_KEY_TX, NO_KEY_TRANSMIT)
- **SM_STATE** (AUTH_KEY_TX, KEY_TRANSMIT)
- **SM_STEP** (AUTH_KEY_TX)
- **SM_STATE** (KEY_RX, NO_KEY_RECEIVE)
- **SM_STATE** (KEY_RX, KEY_RECEIVE)
- **SM_STEP** (KEY_RX)
- **SM_STATE** (CTRL_DIR, FORCE_BOTH)
- **SM_STATE** (CTRL_DIR, IN_OR_BOTH)
- **SM_STEP** (CTRL_DIR)
- struct eapol_state_machine * **eapol_auth_alloc** (struct eapol_authenticator *eapol, const u8 *addr, int preauth, struct sta_info *sta)
- void **eapol_auth_free** (struct eapol_state_machine *sm)
- void eapol_auth_step (struct eapol_state_machine *sm)

  Advance EAPOL state machines.

- void **eapol_auth_initialize** (struct eapol_state_machine *sm)
- int **eapol_auth_eap_pending_cb** (struct eapol_state_machine *sm, void *ctx)
- struct eapol_authenticator * **eapol_auth_init** (struct eapol_auth_config *conf, struct eapol_auth_cb *cb)
- void **eapol_auth_deinit** (struct eapol_authenticator *eapol)

15.20.1 Detailed Description

hostapd / IEEE 802.1X-2004 Authenticator - EAPOL state machine

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.20.2 Function Documentation

15.20.2.1 void eapol_auth_step (struct eapol_state_machine * *sm*)

Advance EAPOL state machines.

Parameters:

sm EAPOL state machine

This function is called to advance EAPOL state machines after any change that could affect their state.

15.21 hostapd/eapol_sm.h File Reference

hostapd / IEEE 802.1X-2004 Authenticator - EAPOL state machine `#include "defs.h"`

`#include "radius/radius.h"`

Data Structures

- struct eapol_auth_config
- struct eapol_auth_cb
- struct eapol_authenticator

  Global EAPOL authenticator data.

- struct eapol_state_machine

  Per-Supplicant Authenticator state machines.

Defines

- #define **AUTH_PAE_DEFAULT_quietPeriod** 60
- #define **AUTH_PAE_DEFAULT_reAuthMax** 2
- #define **BE_AUTH_DEFAULT_serverTimeout** 30
- #define **EAPOL_SM_PREAUTH** BIT(0)
- #define **EAPOL_SM_WAIT_START** BIT(1)

Typedefs

- typedef unsigned int **Counter**

Enumerations

- enum **PortTypes** { **ForceUnauthorized** = 1, **ForceAuthorized** = 3, **Auto** = 2 }
- enum **PortState** { **Unauthorized** = 2, **Authorized** = 1 }
- enum **ControlledDirection** { **Both** = 0, **In** = 1 }
- enum **eapol_logger_level** { **EAPOL_LOGGER_DEBUG**, **EAPOL_LOGGER_INFO**, **EAPOL_LOGGER_WARNING** }

Functions

- struct eapol_authenticator * **eapol_auth_init** (struct eapol_auth_config *conf, struct eapol_auth_cb *cb)
- void **eapol_auth_deinit** (struct eapol_authenticator *eapol)
- struct eapol_state_machine * **eapol_auth_alloc** (struct eapol_authenticator *eapol, const u8 *addr, int preauth, struct sta_info *sta)
- void **eapol_auth_free** (struct eapol_state_machine *sm)
- void eapol_auth_step (struct eapol_state_machine *sm)

  Advance EAPOL state machines.

- void **eapol_auth_initialize** (struct eapol_state_machine *sm)

- void **eapol_auth_dump_state** (FILE *f, const char *prefix, struct eapol_state_machine *sm)
- int **eapol_auth_eap_pending_cb** (struct eapol_state_machine *sm, void *ctx)

15.21.1 Detailed Description

hostapd / IEEE 802.1X-2004 Authenticator - EAPOL state machine

Copyright

Copyright (c) 2002-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.21.2 Function Documentation

15.21.2.1 void eapol_auth_step (struct eapol_state_machine * *sm*)

Advance EAPOL state machines.

Parameters:

sm EAPOL state machine

This function is called to advance EAPOL state machines after any change that could affect their state.

15.22 hostapd/hostapd.c File Reference

hostapd / Initialization and configuration `#include "includes.h"`

```
#include "eloop.h"
#include "hostapd.h"
#include "ieee802_1x.h"
#include "beacon.h"
#include "hw_features.h"
#include "accounting.h"
#include "eapol_sm.h"
#include "iapp.h"
#include "ieee802_11_defs.h"
#include "ieee802_11_auth.h"
#include "sta_info.h"
#include "ap_list.h"
#include "driver_i.h"
#include "radius/radius_client.h"
#include "radius/radius_server.h"
#include "wpa.h"
#include "preauth.h"
#include "vlan_init.h"
#include "ctrl_iface.h"
#include "tls.h"
#include "eap_server/eap_sim_db.h"
#include "eap_server/eap.h"
#include "eap_server/tncs.h"
#include "version.h"
#include "l2_packet/l2_packet.h"
#include "wps_hostapd.h"
#include "tkip_countermeasures.h"
```

Data Structures

- struct wpa_auth_iface_iter_data

Functions

- int **hostapd_reload_config** (struct hostapd_iface *iface)
- int **handle_reload_iface** (struct hostapd_iface *iface, void *ctx)

- int **hostapd_setup_interface_complete** (struct hostapd_iface *iface, int err)
- int hostapd_setup_interface (struct hostapd_iface *iface)

  Setup of an interface.

- struct hostapd_data * hostapd_alloc_bss_data (struct hostapd_iface *hapd_iface, struct hostapd_config *conf, struct hostapd_bss_config *bss)

  Allocate and initialize per-BSS data.

- void **hostapd_interface_deinit** (struct hostapd_iface *iface)
- int **hostapd_register_probereq_cb** (struct hostapd_data *hapd, void(*cb)(void *ctx, const u8 *sa, const u8 *ie, size_t ie_len), void *ctx)

Variables

- int **wpa_debug_level**

15.22.1 Detailed Description

hostapd / Initialization and configuration

Copyright

Copyright (c) 2002-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.22.2 Function Documentation

15.22.2.1 struct hostapd_data* hostapd_alloc_bss_data (struct hostapd_iface * *hapd_iface*, struct hostapd_config * *conf*, struct hostapd_bss_config * *bss*) [read]

Allocate and initialize per-BSS data.

Parameters:

hapd_iface Pointer to interface data

conf Pointer to per-interface configuration

bss Pointer to per-BSS configuration for this BSS

Returns:

Pointer to allocated BSS data

This function is used to allocate per-BSS data structure. This data will be freed after hostapd_cleanup() is called for it during interface deinitialization.

15.22.2.2 int hostapd_setup_interface (struct hostapd_iface * *iface*)

Setup of an interface.

Parameters:

iface Pointer to interface data.

Returns:

0 on success, -1 on failure

Initializes the driver interface, validates the configuration, and sets driver parameters based on the configuration. Flushes old stations, sets the channel, encryption, beacons, and WDS links based on the configuration.

15.23 hostapd/hostapd.h File Reference

hostapd / Initialization and configuration Host AP kernel driver `#include "common.h"`

Data Structures

- struct hostapd_probereq_cb
- struct hostapd_data

  hostapd per-BSS data structure

- struct hostapd_iface

  hostapd per-interface data structure

Defines

- #define **MAX_VLAN_ID** 4094
- #define **STA_HASH_SIZE** 256
- #define **STA_HASH**(sta) (sta[5])
- #define **AID_WORDS** ((2008 + 31) / 32)

Functions

- int **hostapd_reload_config** (struct hostapd_iface *iface)
- struct hostapd_data * hostapd_alloc_bss_data (struct hostapd_iface *hapd_iface, struct hostapd_config *conf, struct hostapd_bss_config *bss)

  Allocate and initialize per-BSS data.

- int hostapd_setup_interface (struct hostapd_iface *iface)

  Setup of an interface.

- int **hostapd_setup_interface_complete** (struct hostapd_iface *iface, int err)
- void **hostapd_interface_deinit** (struct hostapd_iface *iface)
- int **handle_reload_iface** (struct hostapd_iface *iface, void *ctx)
- int **handle_dump_state_iface** (struct hostapd_iface *iface, void *ctx)
- int **hostapd_for_each_interface** (int(*cb)(struct hostapd_iface *iface, void *ctx), void *ctx)
- int **hostapd_register_probereq_cb** (struct hostapd_data *hapd, void(*cb)(void *ctx, const u8 *sa, const u8 *ie, size_t ie_len), void *ctx)

15.23.1 Detailed Description

hostapd / Initialization and configuration Host AP kernel driver

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi> Copyright (c) 2007-2008, Intel Corporation

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

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

See README and COPYING for more details.

15.23.2 Function Documentation

15.23.2.1 struct hostapd_data* hostapd_alloc_bss_data (struct hostapd_iface * *hapd_iface*, struct hostapd_config * *conf*, struct hostapd_bss_config * *bss*) [read]

Allocate and initialize per-BSS data.

Parameters:

hapd_iface Pointer to interface data

conf Pointer to per-interface configuration

bss Pointer to per-BSS configuration for this BSS

Returns:

Pointer to allocated BSS data

This function is used to allocate per-BSS data structure. This data will be freed after hostapd_cleanup() is called for it during interface deinitialization.

15.23.2.2 int hostapd_setup_interface (struct hostapd_iface * *iface*)

Setup of an interface.

Parameters:

iface Pointer to interface data.

Returns:

0 on success, -1 on failure

Initializes the driver interface, validates the configuration, and sets driver parameters based on the configuration. Flushes old stations, sets the channel, encryption, beacons, and WDS links based on the configuration.

15.24 hostapd/hostapd_cli.c File Reference

hostapd - command line interface for hostapd daemon `#include "includes.h"`

```
#include <dirent.h>
#include "wpa_ctrl.h"
#include "common.h"
#include "version.h"
```

Data Structures

- struct hostapd_cli_cmd

Functions

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

15.24.1 Detailed Description

hostapd - command line interface for hostapd daemon

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.25 hostapd/hw_features.c File Reference

hostapd / Hardware feature query and different modes `#include "includes.h"`

```
#include "hostapd.h"
#include "ieee802_11_defs.h"
#include "ieee802_11_common.h"
#include "eloop.h"
#include "hw_features.h"
#include "driver_i.h"
#include "config.h"
```

Functions

- void **hostapd_free_hw_features** (struct hostapd_hw_modes *hw_features, size_t num_hw_features)
- int **hostapd_get_hw_features** (struct hostapd_iface *iface)
- int **hostapd_check_ht_capab** (struct hostapd_iface *iface)
- int hostapd_select_hw_mode (struct hostapd_iface *iface)

  Select the hardware mode.

- const char * **hostapd_hw_mode_txt** (int mode)
- int **hostapd_hw_get_freq** (struct hostapd_data *hapd, int chan)
- int **hostapd_hw_get_channel** (struct hostapd_data *hapd, int freq)

15.25.1 Detailed Description

hostapd / Hardware feature query and different modes

Copyright

Copyright 2002-2003, Instant802 Networks, Inc. Copyright 2005-2006, Devicescape Software, Inc. Copyright (c) 2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.25.2 Function Documentation

15.25.2.1 int hostapd_select_hw_mode (struct hostapd_iface * *iface*)

Select the hardware mode.

Parameters:

iface Pointer to interface data.

Returns:

0 on success, -1 on failure

Sets up the hardware mode, channel, rates, and passive scanning based on the configuration.

15.26 hostapd/hw_features.h File Reference

hostapd / Hardware feature query and different modes

15.26.1 Detailed Description

hostapd / Hardware feature query and different modes

Copyright

15.27 hostapd/iapp.c File Reference

hostapd / IEEE 802.11F-2003 Inter-Access Point Protocol (IAPP) `#include "includes.h"`

```
#include <net/if.h>
#include <sys/ioctl.h>
#include <netpacket/packet.h>
#include "hostapd.h"
#include "config.h"
#include "ieee802_11.h"
#include "iapp.h"
#include "eloop.h"
#include "sta_info.h"
```

Data Structures

- struct iapp_hdr
- struct iapp_add_notify
- struct iapp_layer2_update
- struct iapp_move_notify
- struct iapp_move_response
- struct iapp_cache_notify
- struct iapp_cache_response
- struct iapp_send_security_block
- struct iapp_ack_security_block
- struct iapp_data

Defines

- #define **IAPP_MULTICAST** "224.0.1.178"
- #define **IAPP_UDP_PORT** 3517
- #define **IAPP_TCP_PORT** 3517
- #define **IAPP_VERSION** 0

Enumerations

- enum **IAPP_COMMAND** {

  IAPP_CMD_ADD_notify = 0, **IAPP_CMD_MOVE_notify** = 1, **IAPP_CMD_MOVE_response** = 2, **IAPP_CMD_Send_Security_Block** = 3,

  IAPP_CMD_ACK_Security_Block = 4, **IAPP_CMD_CACHE_notify** = 5, **IAPP_CMD_CACHE_response** = 6 }
- enum { **IAPP_MOVE_SUCCESSFUL** = 0, **IAPP_MOVE_DENIED** = 1, **IAPP_MOVE_STALE_MOVE** = 2 }
- enum { **IAPP_CACHE_SUCCESSFUL** = 0, **IAPP_CACHE_STALE_CACHE** = 1 }

Functions

- void iapp_new_station (struct iapp_data *iapp, struct sta_info *sta)

  IAPP processing for a new STA.

- struct iapp_data * **iapp_init** (struct hostapd_data *hapd, const char *iface)
- void **iapp_deinit** (struct iapp_data *iapp)
- int **iapp_reconfig** (struct hostapd_data *hapd, struct hostapd_config *oldconf, struct hostapd_bss_config *oldbss)

Variables

- struct iapp_hdr **packed**

15.27.1 Detailed Description

hostapd / IEEE 802.11F-2003 Inter-Access Point Protocol (IAPP)

Copyright

Copyright (c) 2002-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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: IEEE 802.11F-2003 was a experimental use specification. It has expired and IEEE has withdrawn it. In other words, it is likely better to look at using some other mechanism for AP-to-AP communication than extending the implementation here.

15.27.2 Function Documentation

15.27.2.1 void iapp_new_station (struct iapp_data * *iapp*, struct sta_info * *sta*)

IAPP processing for a new STA.

Parameters:

iapp IAPP data

sta The associated station

15.28 hostapd/iapp.h File Reference

hostapd / IEEE 802.11F-2003 Inter-Access Point Protocol (IAPP)

15.28.1 Detailed Description

hostapd / IEEE 802.11F-2003 Inter-Access Point Protocol (IAPP)

Copyright

15.29 hostapd/ieee802_11.c File Reference

hostapd / IEEE 802.11 Management `#include "includes.h"`

```
#include <net/if.h>
#include "eloop.h"
#include "hostapd.h"
#include "ieee802_11.h"
#include "beacon.h"
#include "hw_features.h"
#include "radius/radius.h"
#include "radius/radius_client.h"
#include "ieee802_11_auth.h"
#include "sta_info.h"
#include "crypto.h"
#include "ieee802_1x.h"
#include "wpa.h"
#include "wme.h"
#include "ap_list.h"
#include "accounting.h"
#include "driver_i.h"
#include "mlme.h"
#include "wpa_ctrl.h"
```

Functions

- u8 * **hostapd_eid_supp_rates** (struct hostapd_data *hapd, u8 *eid)
- u8 * **hostapd_eid_ext_supp_rates** (struct hostapd_data *hapd, u8 *eid)
- u8 * **hostapd_eid_ht_capabilities_info** (struct hostapd_data *hapd, u8 *eid)
- u8 * **hostapd_eid_ht_operation** (struct hostapd_data *hapd, u8 *eid)
- u16 **hostapd_own_capab_info** (struct hostapd_data *hapd, struct sta_info *sta, int probe)
- void **ieee802_11_print_ssid** (char *buf, const u8 *ssid, u8 len)
- void ieee802_11_send_deauth (struct hostapd_data *hapd, u8 *addr, u16 reason)

  Send Deauthentication frame.

- void ieee802_11_mgmt (struct hostapd_data *hapd, u8 *buf, size_t len, u16 stype, struct hostapd_frame_info *fi)

  process incoming IEEE 802.11 management frames

- void ieee802_11_mgmt_cb (struct hostapd_data *hapd, u8 *buf, size_t len, u16 stype, int ok)

  Process management frame TX status callback.

- int **ieee802_11_get_mib** (struct hostapd_data *hapd, char *buf, size_t buflen)

- int **ieee802_11_get_mib_sta** (struct hostapd_data *hapd, struct sta_info *sta, char *buf, size_t buflen)

15.29.1 Detailed Description

hostapd / IEEE 802.11 Management

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi> Copyright (c) 2007-2008, Intel Corporation

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

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

See README and COPYING for more details.

15.29.2 Function Documentation

15.29.2.1 void ieee802_11_mgmt (struct hostapd_data * *hapd*, u8 * *buf*, size_t *len*, u16 *stype*, struct hostapd_frame_info * *fi*)

process incoming IEEE 802.11 management frames

Parameters:

hapd hostapd BSS data structure (the BSS to which the management frame was sent to)

buf management frame data (starting from IEEE 802.11 header)

len length of frame data in octets

stype management frame subtype from frame control field

fi meta data about received frame (signal level, etc.)

Process all incoming IEEE 802.11 management frames. This will be called for each frame received from the kernel driver through wlan::ap interface. In addition, it can be called to re-inserted pending frames (e.g., when using external RADIUS server as an MAC ACL).

15.29.2.2 void ieee802_11_mgmt_cb (struct hostapd_data * *hapd*, u8 * *buf*, size_t *len*, u16 *stype*, int *ok*)

Process management frame TX status callback.

Parameters:

hapd hostapd BSS data structure (the BSS from which the management frame was sent from)

buf management frame data (starting from IEEE 802.11 header)

len length of frame data in octets

stype management frame subtype from frame control field

ok Whether the frame was ACK'ed

15.29.2.3 void ieee802_11_send_deauth (struct hostapd_data * *hapd*, u8 * *addr*, u16 *reason*)

Send Deauthentication frame.

Parameters:

hapd hostapd BSS data

addr Address of the destination STA

reason Reason code for Deauthentication

15.30 hostapd/ieee802_11.h File Reference

hostapd / IEEE 802.11 Management `#include "ieee802_11_defs.h"`

`#include "ieee802_11_common.h"`

Functions

- void ieee802_11_send_deauth (struct hostapd_data *hapd, u8 *addr, u16 reason)

  Send Deauthentication frame.

- void ieee802_11_mgmt (struct hostapd_data *hapd, u8 *buf, size_t len, u16 stype, struct hostapd_frame_info *fi)

  process incoming IEEE 802.11 management frames

- void ieee802_11_mgmt_cb (struct hostapd_data *hapd, u8 *buf, size_t len, u16 stype, int ok)

  Process management frame TX status callback.

- void **ieee802_11_print_ssid** (char *buf, const u8 *ssid, u8 len)
- u16 **hostapd_own_capab_info** (struct hostapd_data *hapd, struct sta_info *sta, int probe)
- u8 * **hostapd_eid_supp_rates** (struct hostapd_data *hapd, u8 *eid)
- u8 * **hostapd_eid_ext_supp_rates** (struct hostapd_data *hapd, u8 *eid)
- u8 * **hostapd_eid_ht_capabilities_info** (struct hostapd_data *hapd, u8 *eid)
- u8 * **hostapd_eid_ht_operation** (struct hostapd_data *hapd, u8 *eid)
- int **hostapd_ht_operation_update** (struct hostapd_iface *iface)
- void **ieee802_11_send_sa_query_req** (struct hostapd_data *hapd, const u8 *addr, const u8 *trans_id)

15.30.1 Detailed Description

hostapd / IEEE 802.11 Management

Copyright

Copyright (c) 2002-2006, Jouni Malinen <j@w1.fi> Copyright (c) 2007-2008, Intel Corporation

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

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

See README and COPYING for more details.

15.30.2 Function Documentation

15.30.2.1 void ieee802_11_mgmt (struct hostapd_data * *hapd*, u8 * *buf*, size_t *len*, u16 *stype*, struct hostapd_frame_info * *fi*)

process incoming IEEE 802.11 management frames

Parameters:

hapd hostapd BSS data structure (the BSS to which the management frame was sent to)

buf management frame data (starting from IEEE 802.11 header)

len length of frame data in octets

stype management frame subtype from frame control field

fi meta data about received frame (signal level, etc.)

Process all incoming IEEE 802.11 management frames. This will be called for each frame received from the kernel driver through wlan::ap interface. In addition, it can be called to re-inserted pending frames (e.g., when using external RADIUS server as an MAC ACL).

15.30.2.2 void ieee802_11_mgmt_cb (struct hostapd_data * *hapd*, u8 * *buf*, size_t *len*, u16 *stype*, int *ok*)

Process management frame TX status callback.

Parameters:

hapd hostapd BSS data structure (the BSS from which the management frame was sent from)

buf management frame data (starting from IEEE 802.11 header)

len length of frame data in octets

stype management frame subtype from frame control field

ok Whether the frame was ACK'ed

15.30.2.3 void ieee802_11_send_deauth (struct hostapd_data * *hapd*, u8 * *addr*, u16 *reason*)

Send Deauthentication frame.

Parameters:

hapd hostapd BSS data

addr Address of the destination STA

reason Reason code for Deauthentication

15.31 hostapd/ieee802_11_auth.c File Reference

hostapd / IEEE 802.11 authentication (ACL) `#include "includes.h"`

```
#include "hostapd.h"
#include "config.h"
#include "ieee802_11.h"
#include "ieee802_11_auth.h"
#include "radius/radius.h"
#include "radius/radius_client.h"
#include "eloop.h"
```

Data Structures

- struct hostapd_cached_radius_acl
- struct hostapd_acl_query_data

Defines

- #define **RADIUS_ACL_TIMEOUT** 30

Functions

- int hostapd_allowed_address (struct hostapd_data *hapd, const u8 *addr, const u8 *msg, size_t len, u32 *session_timeout, u32 *acct_interim_interval, int *vlan_id)

  Check whether a specified STA can be authenticated.

- int hostapd_acl_init (struct hostapd_data *hapd)
- void hostapd_acl_deinit (struct hostapd_data *hapd)

  Deinitialize IEEE 802.11 ACL.

- int **hostapd_acl_reconfig** (struct hostapd_data *hapd, struct hostapd_config *oldconf)

15.31.1 Detailed Description

hostapd / IEEE 802.11 authentication (ACL)

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

Access control list for IEEE 802.11 authentication can uses statically configured ACL from configuration files or an external RADIUS server. Results from external RADIUS queries are cached to allow faster authentication frame processing.

15.31.2 Function Documentation

15.31.2.1 void hostapd_acl_deinit (struct hostapd_data * *hapd*)

Deinitialize IEEE 802.11 ACL.

Parameters:

hapd hostapd BSS data

15.31.2.2 int hostapd_acl_init (struct hostapd_data * *hapd*)

hostapd_acl_init: Initialize IEEE 802.11 ACL : hostapd BSS data Returns: 0 on success, -1 on failure

15.31.2.3 int hostapd_allowed_address (struct hostapd_data * *hapd*, const u8 * *addr*, const u8 * *msg*, size_t *len*, u32 * *session_timeout*, u32 * *acct_interim_interval*, int * *vlan_id*)

Check whether a specified STA can be authenticated.

Parameters:

hapd hostapd BSS data

addr MAC address of the STA

msg Authentication message

len Length of msg in octets

session_timeout Buffer for returning session timeout (from RADIUS)

acct_interim_interval Buffer for returning account interval (from RADIUS)

vlan_id Buffer for returning VLAN ID

Returns:

HOSTAPD_ACL_ACCEPT, HOSTAPD_ACL_REJECT, or HOSTAPD_ACL_PENDING

15.32 hostapd/ieee802_11_auth.h File Reference

hostapd / IEEE 802.11 authentication (ACL)

Enumerations

- enum { **HOSTAPD_ACL_REJECT** = 0, **HOSTAPD_ACL_ACCEPT** = 1, **HOSTAPD_ACL_PENDING** = 2, **HOSTAPD_ACL_ACCEPT_TIMEOUT** = 3 }

Functions

- int hostapd_allowed_address (struct hostapd_data *hapd, const u8 *addr, const u8 *msg, size_t len, u32 *session_timeout, u32 *acct_interim_interval, int *vlan_id)

  Check whether a specified STA can be authenticated.

- int hostapd_acl_init (struct hostapd_data *hapd)
- void hostapd_acl_deinit (struct hostapd_data *hapd)

  Deinitialize IEEE 802.11 ACL.

- int **hostapd_acl_reconfig** (struct hostapd_data *hapd, struct hostapd_config *oldconf)

15.32.1 Detailed Description

hostapd / IEEE 802.11 authentication (ACL)

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.32.2 Function Documentation

15.32.2.1 void hostapd_acl_deinit (struct hostapd_data * *hapd*)

Deinitialize IEEE 802.11 ACL.

Parameters:

hapd hostapd BSS data

15.32.2.2 int hostapd_acl_init (struct hostapd_data * *hapd*)

hostapd_acl_init: Initialize IEEE 802.11 ACL : hostapd BSS data Returns: 0 on success, -1 on failure

15.32.2.3 int hostapd_allowed_address (struct hostapd_data * *hapd*, const u8 * *addr*, const u8 * *msg*, size_t *len*, u32 * *session_timeout*, u32 * *acct_interim_interval*, int * *vlan_id*)

Check whether a specified STA can be authenticated.

Parameters:

hapd hostapd BSS data

addr MAC address of the STA

msg Authentication message

len Length of msg in octets

session_timeout Buffer for returning session timeout (from RADIUS)

acct_interim_interval Buffer for returning account interval (from RADIUS)

vlan_id Buffer for returning VLAN ID

Returns:

HOSTAPD_ACL_ACCEPT, HOSTAPD_ACL_REJECT, or HOSTAPD_ACL_PENDING

15.33 hostapd/ieee802_1x.c File Reference

hostapd / IEEE 802.1X-2004 Authenticator `#include "includes.h"`

```
#include "hostapd.h"
#include "ieee802_1x.h"
#include "accounting.h"
#include "radius/radius.h"
#include "radius/radius_client.h"
#include "eapol_sm.h"
#include "md5.h"
#include "crypto.h"
#include "eloop.h"
#include "sta_info.h"
#include "wpa.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "driver_i.h"
#include "hw_features.h"
#include "eap_server/eap.h"
#include "ieee802_11_defs.h"
#include "wpa_ctrl.h"
```

Data Structures

- struct sta_id_search

Functions

- void **ieee802_1x_set_sta_authorized** (struct hostapd_data *hapd, struct sta_info *sta, int authorized)
- void **ieee802_1x_tx_key** (struct hostapd_data *hapd, struct sta_info *sta)
- const char * **radius_mode_txt** (struct hostapd_data *hapd)
- int **radius_sta_rate** (struct hostapd_data *hapd, struct sta_info *sta)
- char * **eap_type_text** (u8 type)
- void ieee802_1x_receive (struct hostapd_data *hapd, const u8 *sa, const u8 *buf, size_t len)

  Process the EAPOL frames from the Supplicant.

- void ieee802_1x_new_station (struct hostapd_data *hapd, struct sta_info *sta)

  Start IEEE 802.1X authentication.

- void **ieee802_1x_free_station** (struct sta_info *sta)
- void **ieee802_1x_abort_auth** (struct hostapd_data *hapd, struct sta_info *sta)

- int **ieee802_1x_init** (struct hostapd_data *hapd)
- void **ieee802_1x_deinit** (struct hostapd_data *hapd)
- int **ieee802_1x_reconfig** (struct hostapd_data *hapd, struct hostapd_config *oldconf, struct hostapd_bss_config *oldbss)
- int **ieee802_1x_tx_status** (struct hostapd_data *hapd, struct sta_info *sta, const u8 *buf, size_t len, int ack)
- u8 * **ieee802_1x_get_identity** (struct eapol_state_machine *sm, size_t *len)
- u8 * **ieee802_1x_get_radius_class** (struct eapol_state_machine *sm, size_t *len, int idx)
- const u8 * **ieee802_1x_get_key** (struct eapol_state_machine *sm, size_t *len)
- void **ieee802_1x_notify_port_enabled** (struct eapol_state_machine *sm, int enabled)
- void **ieee802_1x_notify_port_valid** (struct eapol_state_machine *sm, int valid)
- void **ieee802_1x_notify_pre_auth** (struct eapol_state_machine *sm, int pre_auth)
- int **ieee802_1x_get_mib** (struct hostapd_data *hapd, char *buf, size_t buflen)
- int **ieee802_1x_get_mib_sta** (struct hostapd_data *hapd, struct sta_info *sta, char *buf, size_t buflen)

15.33.1 Detailed Description

hostapd / IEEE 802.1X-2004 Authenticator

Copyright

15.33.2 Function Documentation

15.33.2.1 void ieee802_1x_new_station (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Start IEEE 802.1X authentication.

Parameters:

hapd hostapd BSS data

sta The station

This function is called to start IEEE 802.1X authentication when a new station completes IEEE 802.11 association.

15.33.2.2 void ieee802_1x_receive (struct hostapd_data * *hapd*, const u8 * *sa*, const u8 * *buf*, size_t *len*)

Process the EAPOL frames from the Supplicant.

Parameters:

hapd hostapd BSS data

sa Source address (sender of the EAPOL frame)

buf EAPOL frame

len Length of buf in octets

This function is called for each incoming EAPOL frame from the interface

15.34 hostapd/ieee802_1x.h File Reference

hostapd / IEEE 802.1X-2004 Authenticator

Data Structures

- struct ieee802_1x_eapol_key

Functions

- void ieee802_1x_receive (struct hostapd_data *hapd, const u8 *sa, const u8 *buf, size_t len)

  Process the EAPOL frames from the Supplicant.

- void ieee802_1x_new_station (struct hostapd_data *hapd, struct sta_info *sta)

  Start IEEE 802.1X authentication.

- void **ieee802_1x_free_station** (struct sta_info *sta)
- void **ieee802_1x_tx_key** (struct hostapd_data *hapd, struct sta_info *sta)
- void **ieee802_1x_abort_auth** (struct hostapd_data *hapd, struct sta_info *sta)
- void **ieee802_1x_set_sta_authorized** (struct hostapd_data *hapd, struct sta_info *sta, int authorized)
- void **ieee802_1x_dump_state** (FILE *f, const char *prefix, struct sta_info *sta)
- int **ieee802_1x_init** (struct hostapd_data *hapd)
- void **ieee802_1x_deinit** (struct hostapd_data *hapd)
- int **ieee802_1x_reconfig** (struct hostapd_data *hapd, struct hostapd_config *oldconf, struct hostapd_bss_config *oldbss)
- int **ieee802_1x_tx_status** (struct hostapd_data *hapd, struct sta_info *sta, const u8 *buf, size_t len, int ack)
- u8 * **ieee802_1x_get_identity** (struct eapol_state_machine *sm, size_t *len)
- u8 * **ieee802_1x_get_radius_class** (struct eapol_state_machine *sm, size_t *len, int idx)
- const u8 * **ieee802_1x_get_key** (struct eapol_state_machine *sm, size_t *len)
- void **ieee802_1x_notify_port_enabled** (struct eapol_state_machine *sm, int enabled)
- void **ieee802_1x_notify_port_valid** (struct eapol_state_machine *sm, int valid)
- void **ieee802_1x_notify_pre_auth** (struct eapol_state_machine *sm, int pre_auth)
- int **ieee802_1x_get_mib** (struct hostapd_data *hapd, char *buf, size_t buflen)
- int **ieee802_1x_get_mib_sta** (struct hostapd_data *hapd, struct sta_info *sta, char *buf, size_t buflen)
- void **hostapd_get_ntp_timestamp** (u8 *buf)
- char * **eap_type_text** (u8 type)
- const char * **radius_mode_txt** (struct hostapd_data *hapd)
- int **radius_sta_rate** (struct hostapd_data *hapd, struct sta_info *sta)

Variables

- struct ieee802_1x_eapol_key **packed**

15.34.1 Detailed Description

hostapd / IEEE 802.1X-2004 Authenticator

Copyright

Copyright (c) 2002-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.34.2 Function Documentation

15.34.2.1 void ieee802_1x_new_station (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Start IEEE 802.1X authentication.

Parameters:

hapd hostapd BSS data

sta The station

This function is called to start IEEE 802.1X authentication when a new station completes IEEE 802.11 association.

15.34.2.2 void ieee802_1x_receive (struct hostapd_data * *hapd*, const u8 * *sa*, const u8 * *buf*, size_t *len*)

Process the EAPOL frames from the Supplicant.

Parameters:

hapd hostapd BSS data

sa Source address (sender of the EAPOL frame)

buf EAPOL frame

len Length of buf in octets

This function is called for each incoming EAPOL frame from the interface

15.35 hostapd/main.c File Reference

hostapd / main() `#include "includes.h"`

```
#include <syslog.h>
#include "eloop.h"
#include "hostapd.h"
#include "version.h"
#include "config.h"
#include "tls.h"
#include "eap_server/eap.h"
#include "eap_server/tncs.h"
```

Data Structures

- struct hapd_interfaces

Functions

- int **hostapd_for_each_interface** (int(∗cb)(struct hostapd_iface ∗iface, void ∗ctx), void ∗ctx)
- int **main** (int argc, char ∗argv[ ])

Variables

- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- int **wpa_debug_timestamp**

15.35.1 Detailed Description

hostapd / main()

Copyright

15.36 wpa_supplicant/main.c File Reference

WPA Supplicant / main() function for UNIX like OSes and MinGW. `#include "includes.h"`

`#include "common.h"`

`#include "wpa_supplicant_i.h"`

`#include "driver_i.h"`

Functions

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

Variables

- struct wpa_driver_ops * **wpa_drivers** [ ]

15.36.1 Detailed Description

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

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.37 hostapd/mlme.c File Reference

hostapd / IEEE 802.11 MLME `#include "includes.h"`

```
#include "hostapd.h"
#include "ieee802_11.h"
#include "wpa.h"
#include "sta_info.h"
#include "mlme.h"
```

Functions

- void mlme_authenticate_indication (struct hostapd_data *hapd, struct sta_info *sta)

  Report the establishment of an authentication.

- void mlme_deauthenticate_indication (struct hostapd_data *hapd, struct sta_info *sta, u16 reason_code)

  Report the invalidation of an.

- void mlme_associate_indication (struct hostapd_data *hapd, struct sta_info *sta)

  Report the establishment of an association with.

- void mlme_reassociate_indication (struct hostapd_data *hapd, struct sta_info *sta)

  Report the establishment of an reassociation.

- void mlme_disassociate_indication (struct hostapd_data *hapd, struct sta_info *sta, u16 reason_code)

  Report disassociation with a specific peer.

- void **mlme_michaelmicfailure_indication** (struct hostapd_data *hapd, const u8 *addr)
- void **mlme_deletekeys_request** (struct hostapd_data *hapd, struct sta_info *sta)

15.37.1 Detailed Description

hostapd / IEEE 802.11 MLME

Copyright

Copyright 2003-2006, Jouni Malinen <j@w1.fi> Copyright 2003-2004, Instant802 Networks, Inc. Copyright 2005-2006, Devicescape Software, Inc.

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

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

See README and COPYING for more details.

15.37.2 Function Documentation

15.37.2.1 void mlme_associate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Report the establishment of an association with. a specific peer MAC entity

Parameters:

hapd BSS data

sta peer STA data

MLME calls this function as a result of the establishment of an association with a specific peer MAC entity that resulted from an association procedure that was initiated by that specific peer MAC entity.

PeerSTAAddress = sta->addr

15.37.2.2 void mlme_authenticate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Report the establishment of an authentication. relationship with a specific peer MAC entity

Parameters:

hapd BSS data

sta peer STA data

MLME calls this function as a result of the establishment of an authentication relationship with a specific peer MAC entity that resulted from an authentication procedure that was initiated by that specific peer MAC entity.

PeerSTAAddress = sta->addr AuthenticationType = sta->auth_alg (WLAN_AUTH_OPEN / WLAN_-AUTH_SHARED_KEY)

15.37.2.3 void mlme_deauthenticate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*, u16 *reason_code*)

Report the invalidation of an. authentication relationship with a specific peer MAC entity

Parameters:

hapd BSS data

sta Peer STA data

reason_code ReasonCode from Deauthentication frame

MLME calls this function as a result of the invalidation of an authentication relationship with a specific peer MAC entity.

PeerSTAAddress = sta->addr

15.37.2.4 void mlme_disassociate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*, u16 *reason_code*)

Report disassociation with a specific peer. MAC entity

Parameters:

hapd BSS data

sta Peer STA data

reason_code ReasonCode from Disassociation frame

MLME calls this function as a result of the invalidation of an association relationship with a specific peer MAC entity.

PeerSTAAddress = sta->addr

15.37.2.5 void mlme_reassociate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Report the establishment of an reassociation. with a specific peer MAC entity

Parameters:

hapd BSS data

sta peer STA data

MLME calls this function as a result of the establishment of an reassociation with a specific peer MAC entity that resulted from a reassociation procedure that was initiated by that specific peer MAC entity.

PeerSTAAddress = sta->addr

sta->previous_ap contains the "Current AP" information from ReassocReq.

15.38 wpa_supplicant/mlme.c File Reference

WPA Supplicant - Client mode MLME. `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "config_ssid.h"
#include "wpa_supplicant_i.h"
#include "notify.h"
#include "driver_i.h"
#include "wpa.h"
#include "ieee802_11_defs.h"
#include "ieee802_11_common.h"
#include "mlme.h"
```

Data Structures

- struct ieee80211_sta_bss

Defines

- #define **IEEE80211_AUTH_TIMEOUT** (200)
- #define **IEEE80211_AUTH_MAX_TRIES** 3
- #define **IEEE80211_ASSOC_TIMEOUT** (200)
- #define **IEEE80211_ASSOC_MAX_TRIES** 3
- #define **IEEE80211_MONITORING_INTERVAL** (2000)
- #define **IEEE80211_PROBE_INTERVAL** (60000)
- #define **IEEE80211_RETRY_AUTH_INTERVAL** (1000)
- #define **IEEE80211_SCAN_INTERVAL** (2000)
- #define **IEEE80211_SCAN_INTERVAL_SLOW** (15000)
- #define **IEEE80211_IBSS_JOIN_TIMEOUT** (20000)
- #define **IEEE80211_PROBE_DELAY** (33)
- #define **IEEE80211_CHANNEL_TIME** (33)
- #define **IEEE80211_PASSIVE_CHANNEL_TIME** (200)
- #define **IEEE80211_SCAN_RESULT_EXPIRE** (10000)
- #define **IEEE80211_IBSS_MERGE_INTERVAL** (30000)
- #define **IEEE80211_IBSS_INACTIVITY_LIMIT** (60000)
- #define **IEEE80211_IBSS_MAX_STA_ENTRIES** 128
- #define **IEEE80211_FC**(type, stype) host_to_le16((type << 2) | (stype << 4))
- #define **IEEE80211_MAX_SUPP_RATES** 32

Functions

- int **ieee80211_sta_get_ssid** (struct wpa_supplicant *wpa_s, u8 *ssid, size_t *len)
- int **ieee80211_sta_associate** (struct wpa_supplicant *wpa_s, struct wpa_driver_associate_params *params)

- int **ieee80211_sta_req_scan** (struct wpa_supplicant *wpa_s, const u8 *ssid, size_t ssid_len)
- struct wpa_scan_results * **ieee80211_sta_get_scan_results** (struct wpa_supplicant *wpa_s)
- int **ieee80211_sta_deauthenticate** (struct wpa_supplicant *wpa_s, u16 reason)
- int **ieee80211_sta_disassociate** (struct wpa_supplicant *wpa_s, u16 reason)
- void **ieee80211_sta_rx** (struct wpa_supplicant *wpa_s, const u8 *buf, size_t len, struct ieee80211_rx_status *rx_status)
- void **ieee80211_sta_free_hw_features** (struct hostapd_hw_modes *hw_features, size_t num_hw_features)
- int **ieee80211_sta_init** (struct wpa_supplicant *wpa_s)
- void **ieee80211_sta_deinit** (struct wpa_supplicant *wpa_s)
- int **ieee80211_sta_set_probe_req_ie** (struct wpa_supplicant *wpa_s, const u8 *ies, size_t ies_len)

15.38.1 Detailed Description

WPA Supplicant - Client mode MLME.

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi> Copyright (c) 2004, Instant802 Networks, Inc. Copyright (c) 2005-2006, Devicescape Software, Inc.

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

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

See README and COPYING for more details.

15.39 hostapd/mlme.h File Reference

hostapd / IEEE 802.11 MLME

Functions

- void mlme_authenticate_indication (struct hostapd_data *hapd, struct sta_info *sta)

  Report the establishment of an authentication.

- void mlme_deauthenticate_indication (struct hostapd_data *hapd, struct sta_info *sta, u16 reason_code)

  Report the invalidation of an.

- void mlme_associate_indication (struct hostapd_data *hapd, struct sta_info *sta)

  Report the establishment of an association with.

- void mlme_reassociate_indication (struct hostapd_data *hapd, struct sta_info *sta)

  Report the establishment of an reassociation.

- void mlme_disassociate_indication (struct hostapd_data *hapd, struct sta_info *sta, u16 reason_code)

  Report disassociation with a specific peer.

- void **mlme_michaelmicfailure_indication** (struct hostapd_data *hapd, const u8 *addr)
- void **mlme_deletekeys_request** (struct hostapd_data *hapd, struct sta_info *sta)

15.39.1 Detailed Description

hostapd / IEEE 802.11 MLME

Copyright

Copyright 2003, Jouni Malinen <j@w1.fi> Copyright 2003-2004, Instant802 Networks, Inc. Copyright 2005-2006, Devicescape Software, Inc.

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

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

See README and COPYING for more details.

15.39.2 Function Documentation

15.39.2.1 void mlme_associate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Report the establishment of an association with. a specific peer MAC entity

Parameters:

hapd BSS data

sta peer STA data

MLME calls this function as a result of the establishment of an association with a specific peer MAC entity that resulted from an association procedure that was initiated by that specific peer MAC entity.

PeerSTAAddress = sta->addr

15.39.2.2 void mlme_authenticate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Report the establishment of an authentication. relationship with a specific peer MAC entity

Parameters:

hapd BSS data

sta peer STA data

MLME calls this function as a result of the establishment of an authentication relationship with a specific peer MAC entity that resulted from an authentication procedure that was initiated by that specific peer MAC entity.

PeerSTAAddress = sta->addr AuthenticationType = sta->auth_alg (WLAN_AUTH_OPEN / WLAN_-AUTH_SHARED_KEY)

15.39.2.3 void mlme_deauthenticate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*, u16 *reason_code*)

Report the invalidation of an. authentication relationship with a specific peer MAC entity

Parameters:

hapd BSS data

sta Peer STA data

reason_code ReasonCode from Deauthentication frame

MLME calls this function as a result of the invalidation of an authentication relationship with a specific peer MAC entity.

PeerSTAAddress = sta->addr

15.39.2.4 void mlme_disassociate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*, u16 *reason_code*)

Report disassociation with a specific peer. MAC entity

Parameters:

hapd BSS data

sta Peer STA data

reason_code ReasonCode from Disassociation frame

MLME calls this function as a result of the invalidation of an association relationship with a specific peer MAC entity.

PeerSTAAddress = sta->addr

15.39.2.5 void mlme_reassociate_indication (struct hostapd_data * *hapd*, struct sta_info * *sta*)

Report the establishment of an reassociation. with a specific peer MAC entity

Parameters:

hapd BSS data

sta peer STA data

MLME calls this function as a result of the establishment of an reassociation with a specific peer MAC entity that resulted from a reassociation procedure that was initiated by that specific peer MAC entity.

PeerSTAAddress = sta->addr

sta->previous_ap contains the "Current AP" information from ReassocReq.

15.40 wpa_supplicant/mlme.h File Reference

WPA Supplicant - Client mode MLME.

15.40.1 Detailed Description

WPA Supplicant - Client mode MLME.

Copyright

15.41 hostapd/nt_password_hash.c File Reference

hostapd - Plaintext password to NtPasswordHash `#include "includes.h"`

`#include "common.h"`

`#include "ms_funcs.h"`

Functions

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

15.41.1 Detailed Description

hostapd - Plaintext password to NtPasswordHash

Copyright

Copyright (c) 2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.42 hostapd/peerkey.c File Reference

hostapd - PeerKey for Direct Link Setup (DLS) `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "sha1.h"
#include "sha256.h"
#include "wpa.h"
#include "defs.h"
#include "wpa_auth_i.h"
#include "wpa_auth_ie.h"
```

15.42.1 Detailed Description

hostapd - PeerKey for Direct Link Setup (DLS)

Copyright

Copyright (c) 2006-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.43 src/rsn_supp/peerkey.c File Reference

WPA Supplicant - PeerKey for Direct Link Setup (DLS). `#include "includes.h"`

15.43.1 Detailed Description

WPA Supplicant - PeerKey for Direct Link Setup (DLS).

Copyright

15.44 hostapd/pmksa_cache.c File Reference

hostapd - PMKSA cache for IEEE 802.11i RSN `#include "includes.h"`

```
#include "common.h"
#include "sta_info.h"
#include "config.h"
#include "eloop.h"
#include "sha1.h"
#include "sha256.h"
#include "eapol_sm.h"
#include "pmksa_cache.h"
```

Data Structures

- struct rsn_pmksa_cache

Defines

- #define **PMKID_HASH_SIZE** 128
- #define **PMKID_HASH**(pmkid) (unsigned int) ((pmkid)[0] & 0x7f)

Functions

- void **pmksa_cache_to_eapol_data** (struct rsn_pmksa_cache_entry *entry, struct eapol_state_machine *eapol)
- struct rsn_pmksa_cache_entry * pmksa_cache_auth_add (struct rsn_pmksa_cache *pmksa, const u8 *pmk, size_t pmk_len, const u8 *aa, const u8 *spa, int session_timeout, struct eapol_state_machine *eapol, int akmp)

  Add a PMKSA cache entry.

- struct rsn_pmksa_cache_entry * **pmksa_cache_add_okc** (struct rsn_pmksa_cache *pmksa, const struct rsn_pmksa_cache_entry *old_entry, const u8 *aa, const u8 *pmkid)
- void pmksa_cache_auth_deinit (struct rsn_pmksa_cache *pmksa)

  Free all entries in PMKSA cache.

- struct rsn_pmksa_cache_entry * pmksa_cache_auth_get (struct rsn_pmksa_cache *pmksa, const u8 *spa, const u8 *pmkid)

  Fetch a PMKSA cache entry.

- struct rsn_pmksa_cache_entry * pmksa_cache_get_okc (struct rsn_pmksa_cache *pmksa, const u8 *aa, const u8 *spa, const u8 *pmkid)

  Fetch a PMKSA cache entry using OKC.

- struct rsn_pmksa_cache * pmksa_cache_auth_init (void(*free_cb)(struct rsn_pmksa_cache_entry *entry, void *ctx), void *ctx)

  Initialize PMKSA cache.

15.44.1 Detailed Description

hostapd - PMKSA cache for IEEE 802.11i RSN

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.44.2 Function Documentation

15.44.2.1 struct rsn_pmksa_cache_entry* pmksa_cache_auth_add (struct rsn_pmksa_cache * *pmksa*, const u8 * *pmk*, size_t *pmk_len*, const u8 * *aa*, const u8 * *spa*, int *session_timeout*, struct eapol_state_machine * *eapol*, int *akmp*) [read]

Add a PMKSA cache entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_auth_init()

pmk The new pairwise master key

pmk_len PMK length in bytes, usually PMK_LEN (32)

aa Authenticator address

spa Supplicant address

session_timeout Session timeout

eapol Pointer to EAPOL state machine data

akmp WPA_KEY_MGMT_* used in key derivation

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 Supplicant, this entry will be replaced with the new entry. PMKID will be calculated based on the PMK.

15.44.2.2 void pmksa_cache_auth_deinit (struct rsn_pmksa_cache * *pmksa*)

Free all entries in PMKSA cache.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_auth_init()

15.44.2.3 struct rsn_pmksa_cache_entry* pmksa_cache_auth_get (struct rsn_pmksa_cache * *pmksa*, const u8 * *spa*, const u8 * *pmkid*) [read]

Fetch a PMKSA cache entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_auth_init()

spa Supplicant 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

15.44.2.4 struct rsn_pmksa_cache* pmksa_cache_auth_init (void(*)(struct rsn_pmksa_cache_entry *entry, void *ctx) *free_cb*, void * *ctx*) [read]

Initialize PMKSA cache.

Parameters:

free_cb Callback function to be called when a PMKSA cache entry is freed

ctx Context pointer for free_cb function

Returns:

Pointer to PMKSA cache data or NULL on failure

15.44.2.5 struct rsn_pmksa_cache_entry* pmksa_cache_get_okc (struct rsn_pmksa_cache * *pmksa*, const u8 * *aa*, const u8 * *spa*, const u8 * *pmkid*) [read]

Fetch a PMKSA cache entry using OKC.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_auth_init()

aa Authenticator address

spa Supplicant address

pmkid PMKID

Returns:

Pointer to PMKSA cache entry or NULL if no match was found

Use opportunistic key caching (OKC) to find a PMK for a supplicant.

15.45 src/rsn_supp/pmksa_cache.c File Reference

WPA Supplicant - RSN PMKSA cache. `#include "includes.h"`

```
#include "common.h"
#include "wpa.h"
#include "eloop.h"
#include "sha1.h"
#include "sha256.h"
#include "wpa_i.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "pmksa_cache.h"
```

Data Structures

- struct rsn_pmksa_cache

Functions

- struct rsn_pmksa_cache_entry * pmksa_cache_add (struct rsn_pmksa_cache *pmksa, const u8 *pmk, size_t pmk_len, const u8 *aa, const u8 *spa, void *network_ctx, int akmp)

  Add a PMKSA cache entry.

- void pmksa_cache_deinit (struct rsn_pmksa_cache *pmksa)

  Free all entries in PMKSA cache.

- struct rsn_pmksa_cache_entry * pmksa_cache_get (struct rsn_pmksa_cache *pmksa, const u8 *aa, const u8 *pmkid)

  Fetch a PMKSA cache entry.

- void pmksa_cache_notify_reconfig (struct rsn_pmksa_cache *pmksa)

  Reconfiguration notification for PMKSA cache.

- struct rsn_pmksa_cache_entry * pmksa_cache_get_opportunistic (struct rsn_pmksa_cache *pmksa, void *network_ctx, const u8 *aa)

  Try to get an opportunistic PMKSA entry.

- struct rsn_pmksa_cache_entry * pmksa_cache_get_current (struct wpa_sm *sm)

  Get the current used PMKSA entry.

- void pmksa_cache_clear_current (struct wpa_sm *sm)

  Clear the current PMKSA entry selection.

- int pmksa_cache_set_current (struct wpa_sm *sm, const u8 *pmkid, const u8 *bssid, void *network_ctx, int try_opportunistic)

  Set the current PMKSA entry selection.

- int pmksa_cache_list (struct rsn_pmksa_cache ∗pmksa, char ∗buf, size_t len)

  Dump text list of entries in PMKSA cache.

- struct rsn_pmksa_cache ∗ pmksa_cache_init (void(∗free_cb)(struct rsn_pmksa_cache_entry ∗entry, void ∗ctx, int replace), void ∗ctx, struct wpa_sm ∗sm)

  Initialize PMKSA cache.

15.45.1 Detailed Description

WPA Supplicant - RSN PMKSA cache.

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.45.2 Function Documentation

15.45.2.1 struct rsn_pmksa_cache_entry∗ pmksa_cache_add (struct rsn_pmksa_cache ∗ *pmksa*, const u8 ∗ *pmk*, size_t *pmk_len*, const u8 ∗ *aa*, const u8 ∗ *spa*, void ∗ *network_ctx*, int *akmp*) [read]

Add a PMKSA cache entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

pmk The new pairwise master key

pmk_len PMK length in bytes, usually PMK_LEN (32)

aa Authenticator address

spa Supplicant address

network_ctx Network configuration context for this PMK

akmp WPA_KEY_MGMT_∗ used in key derivation

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.

15.45.2.2 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()

15.45.2.3 void pmksa_cache_deinit (struct rsn_pmksa_cache * *pmksa*)

Free all entries in PMKSA cache.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

15.45.2.4 struct rsn_pmksa_cache_entry* pmksa_cache_get (struct rsn_pmksa_cache * *pmksa*, const u8 * *aa*, const u8 * *pmkid*) [read]

Fetch a PMKSA cache entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

aa Authenticator address or NULL to match any

pmkid PMKID or NULL to match any

Returns:

Pointer to PMKSA cache entry or NULL if no match was found

15.45.2.5 struct rsn_pmksa_cache_entry* pmksa_cache_get_current (struct wpa_sm * *sm*) [read]

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

15.45.2.6 struct rsn_pmksa_cache_entry* pmksa_cache_get_opportunistic (struct rsn_pmksa_cache * *pmksa*, void * *network_ctx*, const u8 * *aa*) [read]

Try to get an opportunistic PMKSA entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

network_ctx Network configuration context

aa Authenticator address for the new AP

Returns:

Pointer to a new PMKSA cache entry or NULL if not available

Try to create a new PMKSA cache entry opportunistically by guessing that the new AP is sharing the same PMK as another AP that has the same SSID and has already an entry in PMKSA cache.

15.45.2.7 struct rsn_pmksa_cache* pmksa_cache_init (void(*)(struct rsn_pmksa_cache_entry *entry, void *ctx, int replace) *free_cb*, void * *ctx*, struct wpa_sm * *sm*) [read]

Initialize PMKSA cache.

Parameters:

free_cb Callback function to be called when a PMKSA cache entry is freed

ctx Context pointer for free_cb function

sm Pointer to WPA state machine data from wpa_sm_init()

Returns:

Pointer to PMKSA cache data or NULL on failure

15.45.2.8 int pmksa_cache_list (struct rsn_pmksa_cache * *pmksa*, char * *buf*, size_t *len*)

Dump text list of entries in PMKSA cache.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_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.

15.45.2.9 void pmksa_cache_notify_reconfig (struct rsn_pmksa_cache * *pmksa*)

Reconfiguration notification for PMKSA cache.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

Clear references to old data structures when wpa_supplicant is reconfigured.

15.45.2.10 int pmksa_cache_set_current (struct wpa_sm * *sm*, const u8 * *pmkid*, const u8 * *bssid*, void * *network_ctx*, 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

network_ctx Network configuration context

try_opportunistic Whether to allow opportunistic PMKSA caching

Returns:

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

15.46 hostapd/pmksa_cache.h File Reference

hostapd - PMKSA cache for IEEE 802.11i RSN

Data Structures

- struct rsn_pmksa_cache_entry

  PMKSA cache entry.

Functions

- struct rsn_pmksa_cache * pmksa_cache_auth_init (void(*free_cb)(struct rsn_pmksa_cache_entry *entry, void *ctx), void *ctx)

  Initialize PMKSA cache.

- void pmksa_cache_auth_deinit (struct rsn_pmksa_cache *pmksa)

  Free all entries in PMKSA cache.

- struct rsn_pmksa_cache_entry * pmksa_cache_auth_get (struct rsn_pmksa_cache *pmksa, const u8 *spa, const u8 *pmkid)

  Fetch a PMKSA cache entry.

- struct rsn_pmksa_cache_entry * pmksa_cache_get_okc (struct rsn_pmksa_cache *pmksa, const u8 *spa, const u8 *aa, const u8 *pmkid)

  Fetch a PMKSA cache entry using OKC.

- struct rsn_pmksa_cache_entry * pmksa_cache_auth_add (struct rsn_pmksa_cache *pmksa, const u8 *pmk, size_t pmk_len, const u8 *aa, const u8 *spa, int session_timeout, struct eapol_state_machine *eapol, int akmp)

  Add a PMKSA cache entry.

- struct rsn_pmksa_cache_entry * **pmksa_cache_add_okc** (struct rsn_pmksa_cache *pmksa, const struct rsn_pmksa_cache_entry *old_entry, const u8 *aa, const u8 *pmkid)
- void **pmksa_cache_to_eapol_data** (struct rsn_pmksa_cache_entry *entry, struct eapol_state_machine *eapol)

15.46.1 Detailed Description

hostapd - PMKSA cache for IEEE 802.11i RSN

Copyright

15.46.2 Function Documentation

15.46.2.1 struct rsn_pmksa_cache_entry* pmksa_cache_auth_add (struct rsn_pmksa_cache * *pmksa*, const u8 * *pmk*, size_t *pmk_len*, const u8 * *aa*, const u8 * *spa*, int *session_timeout*, struct eapol_state_machine * *eapol*, int *akmp*) [read]

Add a PMKSA cache entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_auth_init()

pmk The new pairwise master key

pmk_len PMK length in bytes, usually PMK_LEN (32)

aa Authenticator address

spa Supplicant address

session_timeout Session timeout

eapol Pointer to EAPOL state machine data

akmp WPA_KEY_MGMT_* used in key derivation

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 Supplicant, this entry will be replaced with the new entry. PMKID will be calculated based on the PMK.

15.46.2.2 void pmksa_cache_auth_deinit (struct rsn_pmksa_cache * *pmksa*)

Free all entries in PMKSA cache.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_auth_init()

15.46.2.3 struct rsn_pmksa_cache_entry* pmksa_cache_auth_get (struct rsn_pmksa_cache * *pmksa*, const u8 * *spa*, const u8 * *pmkid*) [read]

Fetch a PMKSA cache entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_auth_init()

spa Supplicant 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

15.46.2.4 struct rsn_pmksa_cache* pmksa_cache_auth_init (void(*)(struct rsn_pmksa_cache_entry *entry, void *ctx) *free_cb*, void * *ctx*) [read]

Initialize PMKSA cache.

Parameters:

free_cb Callback function to be called when a PMKSA cache entry is freed

ctx Context pointer for free_cb function

Returns:

Pointer to PMKSA cache data or NULL on failure

15.46.2.5 struct rsn_pmksa_cache_entry* pmksa_cache_get_okc (struct rsn_pmksa_cache * *pmksa*, const u8 * *aa*, const u8 * *spa*, const u8 * *pmkid*) [read]

Fetch a PMKSA cache entry using OKC.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_auth_init()

aa Authenticator address

spa Supplicant address

pmkid PMKID

Returns:

Pointer to PMKSA cache entry or NULL if no match was found

Use opportunistic key caching (OKC) to find a PMK for a supplicant.

15.47 src/rsn_supp/pmksa_cache.h File Reference

wpa_supplicant - WPA2/RSN PMKSA cache functions

Data Structures

- struct rsn_pmksa_cache_entry

  PMKSA cache entry.

Functions

- struct rsn_pmksa_cache * pmksa_cache_init (void(*free_cb)(struct rsn_pmksa_cache_entry *entry, void *ctx, int replace), void *ctx, struct wpa_sm *sm)

  Initialize PMKSA cache.

- void pmksa_cache_deinit (struct rsn_pmksa_cache *pmksa)

  Free all entries in PMKSA cache.

- struct rsn_pmksa_cache_entry * pmksa_cache_get (struct rsn_pmksa_cache *pmksa, const u8 *aa, const u8 *pmkid)

  Fetch a PMKSA cache entry.

- int pmksa_cache_list (struct rsn_pmksa_cache *pmksa, char *buf, size_t len)

  Dump text list of entries in PMKSA cache.

- struct rsn_pmksa_cache_entry * pmksa_cache_add (struct rsn_pmksa_cache *pmksa, const u8 *pmk, size_t pmk_len, const u8 *aa, const u8 *spa, void *network_ctx, int akmp)

  Add a PMKSA cache entry.

- void pmksa_cache_notify_reconfig (struct rsn_pmksa_cache *pmksa)

  Reconfiguration notification for PMKSA cache.

- struct rsn_pmksa_cache_entry * pmksa_cache_get_current (struct wpa_sm *sm)

  Get the current used PMKSA entry.

- void pmksa_cache_clear_current (struct wpa_sm *sm)

  Clear the current PMKSA entry selection.

- int pmksa_cache_set_current (struct wpa_sm *sm, const u8 *pmkid, const u8 *bssid, void *network_ctx, int try_opportunistic)

  Set the current PMKSA entry selection.

- struct rsn_pmksa_cache_entry * pmksa_cache_get_opportunistic (struct rsn_pmksa_cache *pmksa, void *network_ctx, const u8 *aa)

  Try to get an opportunistic PMKSA entry.

15.47.1 Detailed Description

wpa_supplicant - WPA2/RSN PMKSA cache functions

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.47.2 Function Documentation

15.47.2.1 struct rsn_pmksa_cache_entry* pmksa_cache_add (struct rsn_pmksa_cache * *pmksa*, const u8 * *pmk*, size_t *pmk_len*, const u8 * *aa*, const u8 * *spa*, void * *network_ctx*, int *akmp*) [read]

Add a PMKSA cache entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

pmk The new pairwise master key

pmk_len PMK length in bytes, usually PMK_LEN (32)

aa Authenticator address

spa Supplicant address

network_ctx Network configuration context for this PMK

akmp WPA_KEY_MGMT_* used in key derivation

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.

15.47.2.2 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()

15.47.2.3 void pmksa_cache_deinit (struct rsn_pmksa_cache * *pmksa*)

Free all entries in PMKSA cache.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

15.47.2.4 struct rsn_pmksa_cache_entry* pmksa_cache_get (struct rsn_pmksa_cache * *pmksa*, const u8 * *aa*, const u8 * *pmkid*) [read]

Fetch a PMKSA cache entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

aa Authenticator address or NULL to match any

pmkid PMKID or NULL to match any

Returns:

Pointer to PMKSA cache entry or NULL if no match was found

15.47.2.5 struct rsn_pmksa_cache_entry* pmksa_cache_get_current (struct wpa_sm * *sm*) [read]

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

15.47.2.6 struct rsn_pmksa_cache_entry* pmksa_cache_get_opportunistic (struct rsn_pmksa_cache * *pmksa*, void * *network_ctx*, const u8 * *aa*) [read]

Try to get an opportunistic PMKSA entry.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

network_ctx Network configuration context

aa Authenticator address for the new AP

Returns:

Pointer to a new PMKSA cache entry or NULL if not available

Try to create a new PMKSA cache entry opportunistically by guessing that the new AP is sharing the same PMK as another AP that has the same SSID and has already an entry in PMKSA cache.

15.47.2.7 struct rsn_pmksa_cache* pmksa_cache_init (void(*)(struct rsn_pmksa_cache_entry *entry, void *ctx, int replace) *free_cb*, void * *ctx*, struct wpa_sm * *sm*) [read]

Initialize PMKSA cache.

Parameters:

free_cb Callback function to be called when a PMKSA cache entry is freed

ctx Context pointer for free_cb function

sm Pointer to WPA state machine data from wpa_sm_init()

Returns:

Pointer to PMKSA cache data or NULL on failure

15.47.2.8 int pmksa_cache_list (struct rsn_pmksa_cache * *pmksa*, char * *buf*, size_t *len*)

Dump text list of entries in PMKSA cache.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_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.

15.47.2.9 void pmksa_cache_notify_reconfig (struct rsn_pmksa_cache * *pmksa*)

Reconfiguration notification for PMKSA cache.

Parameters:

pmksa Pointer to PMKSA cache data from pmksa_cache_init()

Clear references to old data structures when wpa_supplicant is reconfigured.

15.47.2.10 int pmksa_cache_set_current (struct wpa_sm * *sm*, const u8 * *pmkid*, const u8 * *bssid*, void * *network_ctx*, 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

network_ctx Network configuration context

try_opportunistic Whether to allow opportunistic PMKSA caching

Returns:

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

15.48 hostapd/preauth.c File Reference

hostapd - Authenticator for IEEE 802.11i RSN pre-authentication `#include "includes.h"`

15.48.1 Detailed Description

hostapd - Authenticator for IEEE 802.11i RSN pre-authentication

Copyright

15.49 src/rsn_supp/preauth.c File Reference

WPA Supplicant - RSN pre-authentication. `#include "includes.h"`

```
#include "common.h"
#include "wpa.h"
#include "drivers/driver.h"
#include "eloop.h"
#include "l2_packet/l2_packet.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_i.h"
#include "ieee802_11_defs.h"
```

Data Structures

- struct rsn_pmksa_candidate

Defines

- #define **PMKID_CANDIDATE_PRIO_SCAN** 1000

Functions

- void pmksa_candidate_free (struct wpa_sm *sm)
  Free all entries in PMKSA candidate list.
- int rsn_preauth_init (struct wpa_sm *sm, const u8 *dst, struct eap_peer_config *eap_conf)
  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_results *results)
  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.

15.49.1 Detailed Description

WPA Supplicant - RSN pre-authentication.

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.49.2 Function Documentation

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

Add a new PMKSA candidate.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

bssid BSSID (authenticator address) of the candidate

prio Priority (the smaller number, the higher priority)

preauth Whether the candidate AP advertises support for pre-authentication

This function is used to add PMKSA candidates for RSN pre-authentication. It is called from scan result processing and from driver events for PMKSA candidates, i.e., EVENT_PMKID_CANDIDATE events to wpa_supplicant_event().

15.49.2.2 void pmksa_candidate_free (struct wpa_sm * *sm*)

Free all entries in PMKSA candidate list.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

15.49.2.3 void rsn_preauth_candidate_process (struct wpa_sm * *sm*)

Process PMKSA candidates.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

Go through the PMKSA candidates and start pre-authentication if a candidate without an existing PMKSA cache entry is found. Processed candidates will be removed from the list.

15.49.2.4 void rsn_preauth_deinit (struct wpa_sm * *sm*)

Abort RSN pre-authentication.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

This function aborts the current RSN pre-authentication (if one is started) and frees resources allocated for it.

15.49.2.5 int rsn_preauth_get_status (struct wpa_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get pre-authentication status.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()
buf Buffer for status information
buflen Maximum buffer length
verbose Whether to include verbose status information

Returns:

Number of bytes written to buf.

Query WPA2 pre-authentication for status information. This function fills in a text area with current status information. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

15.49.2.6 int rsn_preauth_in_progress (struct wpa_sm * *sm*)

Verify whether pre-authentication is in progress.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

15.49.2.7 int rsn_preauth_init (struct wpa_sm * *sm*, const u8 * *dst*, struct eap_peer_config * *eap_conf*)

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
eap_conf Current EAP 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.

15.49.2.8 void rsn_preauth_scan_results (struct wpa_sm * *sm*, struct wpa_scan_results * *results*)

Process scan results to find PMKSA candidates.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

results Scan results

This functions goes through the scan results and adds all suitable APs (Authenticators) into PMKSA candidate list.

15.50 hostapd/preauth.h File Reference

hostapd - Authenticator for IEEE 802.11i RSN pre-authentication

15.50.1 Detailed Description

hostapd - Authenticator for IEEE 802.11i RSN pre-authentication

Copyright

15.51 src/rsn_supp/preauth.h File Reference

wpa_supplicant - WPA2/RSN pre-authentication functions

Functions

- void pmksa_candidate_free (struct wpa_sm *sm)
  Free all entries in PMKSA candidate list.
- int rsn_preauth_init (struct wpa_sm *sm, const u8 *dst, struct eap_peer_config *eap_conf)
  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_results *results)
  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.

15.51.1 Detailed Description

wpa_supplicant - WPA2/RSN pre-authentication functions

Copyright

15.51.2 Function Documentation

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

Add a new PMKSA candidate.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

bssid BSSID (authenticator address) of the candidate

prio Priority (the smaller number, the higher priority)

preauth Whether the candidate AP advertises support for pre-authentication

This function is used to add PMKSA candidates for RSN pre-authentication. It is called from scan result processing and from driver events for PMKSA candidates, i.e., EVENT_PMKID_CANDIDATE events to wpa_supplicant_event().

15.51.2.2 void pmksa_candidate_free (struct wpa_sm * *sm*)

Free all entries in PMKSA candidate list.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

15.51.2.3 void rsn_preauth_candidate_process (struct wpa_sm * *sm*)

Process PMKSA candidates.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

Go through the PMKSA candidates and start pre-authentication if a candidate without an existing PMKSA cache entry is found. Processed candidates will be removed from the list.

15.51.2.4 void rsn_preauth_deinit (struct wpa_sm * *sm*)

Abort RSN pre-authentication.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

This function aborts the current RSN pre-authentication (if one is started) and frees resources allocated for it.

15.51.2.5 int rsn_preauth_get_status (struct wpa_sm * *sm*, char * *buf*, size_t *buflen*, int *verbose*)

Get pre-authentication status.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

buf Buffer for status information

buflen Maximum buffer length

verbose Whether to include verbose status information

Returns:

Number of bytes written to buf.

Query WPA2 pre-authentication for status information. This function fills in a text area with current status information. If the buffer (buf) is not large enough, status information will be truncated to fit the buffer.

15.51.2.6 int rsn_preauth_in_progress (struct wpa_sm * *sm*)

Verify whether pre-authentication is in progress.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

15.51.2.7 int rsn_preauth_init (struct wpa_sm * *sm*, const u8 * *dst*, struct eap_peer_config * *eap_conf*)

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

eap_conf Current EAP 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.

15.51.2.8 void rsn_preauth_scan_results (struct wpa_sm * *sm*, struct wpa_scan_results * *results*)

Process scan results to find PMKSA candidates.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

results Scan results

This functions goes through the scan results and adds all suitable APs (Authenticators) into PMKSA candidate list.

15.52 hostapd/sta_flags.h File Reference

hostapd - driver interface definition

Defines

- #define **WLAN_STA_AUTH** BIT(0)
- #define **WLAN_STA_ASSOC** BIT(1)
- #define **WLAN_STA_PS** BIT(2)
- #define **WLAN_STA_TIM** BIT(3)
- #define **WLAN_STA_PERM** BIT(4)
- #define **WLAN_STA_AUTHORIZED** BIT(5)
- #define **WLAN_STA_PENDING_POLL** BIT(6)
- #define **WLAN_STA_SHORT_PREAMBLE** BIT(7)
- #define **WLAN_STA_PREAUTH** BIT(8)
- #define **WLAN_STA_WMM** BIT(9)
- #define **WLAN_STA_MFP** BIT(10)
- #define **WLAN_STA_HT** BIT(11)
- #define **WLAN_STA_WPS** BIT(12)
- #define **WLAN_STA_MAYBE_WPS** BIT(13)
- #define **WLAN_STA_NONERP** BIT(31)

15.52.1 Detailed Description

hostapd - driver interface definition

Copyright

15.53 hostapd/sta_info.c File Reference

hostapd / Station table `#include "includes.h"`

```
#include "hostapd.h"
#include "sta_info.h"
#include "eloop.h"
#include "accounting.h"
#include "ieee802_1x.h"
#include "ieee802_11.h"
#include "radius/radius.h"
#include "wpa.h"
#include "preauth.h"
#include "radius/radius_client.h"
#include "driver_i.h"
#include "beacon.h"
#include "hw_features.h"
#include "mlme.h"
#include "vlan_init.h"
```

Functions

- int **ap_for_each_sta** (struct hostapd_data *hapd, int(*cb)(struct hostapd_data *hapd, struct sta_info *sta, void *ctx), void *ctx)
- struct sta_info * **ap_get_sta** (struct hostapd_data *hapd, const u8 *sta)
- void **ap_sta_hash_add** (struct hostapd_data *hapd, struct sta_info *sta)
- void **ap_free_sta** (struct hostapd_data *hapd, struct sta_info *sta)
- void **hostapd_free_stas** (struct hostapd_data *hapd)
- void ap_handle_timer (void *eloop_ctx, void *timeout_ctx)

  Per STA timer handler.

- void **ap_sta_session_timeout** (struct hostapd_data *hapd, struct sta_info *sta, u32 session_timeout)
- void **ap_sta_no_session_timeout** (struct hostapd_data *hapd, struct sta_info *sta)
- struct sta_info * **ap_sta_add** (struct hostapd_data *hapd, const u8 *addr)
- void **ap_sta_disassociate** (struct hostapd_data *hapd, struct sta_info *sta, u16 reason)
- void **ap_sta_deauthenticate** (struct hostapd_data *hapd, struct sta_info *sta, u16 reason)
- int **ap_sta_bind_vlan** (struct hostapd_data *hapd, struct sta_info *sta, int old_vlanid)

15.53.1 Detailed Description

hostapd / Station table

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi> Copyright (c) 2007-2008, Intel Corporation

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

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

See README and COPYING for more details.

15.53.2 Function Documentation

15.53.2.1 void ap_handle_timer (void * *eloop_ctx*, void * *timeout_ctx*)

Per STA timer handler.

Parameters:

eloop_ctx struct hostapd_data *

timeout_ctx struct sta_info *

This function is called to check station activity and to remove inactive stations.

15.54 hostapd/sta_info.h File Reference

hostapd / Station table `#include "sta_flags.h"`

Data Structures

- struct sta_info

Defines

- #define **WLAN_SUPP_RATES_MAX** 32
- #define **AP_MAX_INACTIVITY** (5 * 60)
- #define **AP_DISASSOC_DELAY** (1)
- #define **AP_DEAUTH_DELAY** (1)
- #define **AP_MAX_INACTIVITY_AFTER_DISASSOC** (1 * 30)
- #define **AP_MAX_INACTIVITY_AFTER_DEAUTH** (1 * 5)

Functions

- int **ap_for_each_sta** (struct hostapd_data *hapd, int(*cb)(struct hostapd_data *hapd, struct sta_info *sta, void *ctx), void *ctx)
- struct sta_info * **ap_get_sta** (struct hostapd_data *hapd, const u8 *sta)
- void **ap_sta_hash_add** (struct hostapd_data *hapd, struct sta_info *sta)
- void **ap_free_sta** (struct hostapd_data *hapd, struct sta_info *sta)
- void **hostapd_free_stas** (struct hostapd_data *hapd)
- void ap_handle_timer (void *eloop_ctx, void *timeout_ctx)

  Per STA timer handler.

- void **ap_sta_session_timeout** (struct hostapd_data *hapd, struct sta_info *sta, u32 session_timeout)

- void **ap_sta_no_session_timeout** (struct hostapd_data *hapd, struct sta_info *sta)
- struct sta_info * **ap_sta_add** (struct hostapd_data *hapd, const u8 *addr)
- void **ap_sta_disassociate** (struct hostapd_data *hapd, struct sta_info *sta, u16 reason)
- void **ap_sta_deauthenticate** (struct hostapd_data *hapd, struct sta_info *sta, u16 reason)
- int **ap_sta_bind_vlan** (struct hostapd_data *hapd, struct sta_info *sta, int old_vlanid)
- void **ap_sta_start_sa_query** (struct hostapd_data *hapd, struct sta_info *sta)
- void **ap_sta_stop_sa_query** (struct hostapd_data *hapd, struct sta_info *sta)
- int **ap_check_sa_query_timeout** (struct hostapd_data *hapd, struct sta_info *sta)

15.54.1 Detailed Description

hostapd / Station table

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.54.2 Function Documentation

15.54.2.1 void ap_handle_timer (void * *eloop_ctx*, void * *timeout_ctx*)

Per STA timer handler.

Parameters:

eloop_ctx struct hostapd_data *

timeout_ctx struct sta_info *

This function is called to check station activity and to remove inactive stations.

15.55 hostapd/tkip_countermeasures.c File Reference

hostapd / TKIP countermeasures `#include "includes.h"`

```
#include "hostapd.h"
#include "eloop.h"
#include "driver_i.h"
#include "sta_info.h"
#include "mlme.h"
#include "wpa.h"
#include "ieee802_11_defs.h"
#include "tkip_countermeasures.h"
```

Functions

- void **michael_mic_failure** (struct hostapd_data *hapd, const u8 *addr, int local)

15.55.1 Detailed Description

hostapd / TKIP countermeasures

Copyright

Copyright (c) 2002-2009, Jouni Malinen <j@w1.fi>

15.56 hostapd/tkip_countermeasures.h File Reference

hostapd / TKIP countermeasures

Functions

- void **michael_mic_failure** (struct hostapd_data *hapd, const u8 *addr, int local)

15.56.1 Detailed Description

hostapd / TKIP countermeasures

Copyright

15.57 hostapd/vlan_init.c File Reference

hostapd / VLAN initialization `#include "includes.h"`

`#include "hostapd.h"`

`#include "driver_i.h"`

`#include "vlan_init.h"`

Functions

- int **vlan_setup_encryption_dyn** (struct hostapd_data *hapd, struct hostapd_ssid *mssid, const char *dyn_vlan)
- int **vlan_init** (struct hostapd_data *hapd)
- void **vlan_deinit** (struct hostapd_data *hapd)
- int **vlan_reconfig** (struct hostapd_data *hapd, struct hostapd_config *oldconf, struct hostapd_bss_config *oldbss)
- struct hostapd_vlan * **vlan_add_dynamic** (struct hostapd_data *hapd, struct hostapd_vlan *vlan, int vlan_id)
- int **vlan_remove_dynamic** (struct hostapd_data *hapd, int vlan_id)

15.57.1 Detailed Description

hostapd / VLAN initialization

Copyright

15.58 hostapd/vlan_init.h File Reference

hostapd / VLAN initialization

Functions

- int **vlan_init** (struct hostapd_data *hapd)
- void **vlan_deinit** (struct hostapd_data *hapd)
- int **vlan_reconfig** (struct hostapd_data *hapd, struct hostapd_config *oldconf, struct hostapd_bss_config *oldbss)
- struct hostapd_vlan * **vlan_add_dynamic** (struct hostapd_data *hapd, struct hostapd_vlan *vlan, int vlan_id)
- int **vlan_remove_dynamic** (struct hostapd_data *hapd, int vlan_id)
- int **vlan_setup_encryption_dyn** (struct hostapd_data *hapd, struct hostapd_ssid *mssid, const char *dyn_vlan)

15.58.1 Detailed Description

hostapd / VLAN initialization

Copyright

15.59 hostapd/wme.c File Reference

hostapd / WMM (Wi-Fi Multimedia) `#include "includes.h"`

`#include "hostapd.h"`

`#include "ieee802_11.h"`

`#include "wme.h"`

`#include "sta_info.h"`

`#include "driver_i.h"`

Functions

- u8 * **hostapd_eid_wmm** (struct hostapd_data *hapd, u8 *eid)
- int **hostapd_eid_wmm_valid** (struct hostapd_data *hapd, u8 *eid, size_t len)
- int **hostapd_wmm_sta_config** (struct hostapd_data *hapd, struct sta_info *sta)
- int **wmm_process_tspec** (struct wmm_tspec_element *tspec)
- void **hostapd_wmm_action** (struct hostapd_data *hapd, struct ieee80211_mgmt *mgmt, size_t len)

15.59.1 Detailed Description

hostapd / WMM (Wi-Fi Multimedia)

Copyright

15.60 hostapd/wme.h File Reference

hostapd / WMM (Wi-Fi Multimedia)

Functions

- u8 * **hostapd_eid_wmm** (struct hostapd_data *hapd, u8 *eid)
- int **hostapd_eid_wmm_valid** (struct hostapd_data *hapd, u8 *eid, size_t len)
- void **hostapd_wmm_action** (struct hostapd_data *hapd, struct ieee80211_mgmt *mgmt, size_t len)

- int **wmm_process_tspec** (struct wmm_tspec_element *tspec)

15.60.1 Detailed Description

hostapd / WMM (Wi-Fi Multimedia)

Copyright

15.61 hostapd/wpa.c File Reference

hostapd - IEEE 802.11i-2004 / WPA Authenticator `#include "includes.h"`

```
#include "common.h"
#include "config.h"
#include "eapol_sm.h"
#include "wpa.h"
#include "sha1.h"
#include "sha256.h"
#include "aes_wrap.h"
#include "crypto.h"
#include "eloop.h"
#include "ieee802_11.h"
#include "pmksa_cache.h"
#include "state_machine.h"
#include "wpa_auth_i.h"
#include "wpa_auth_ie.h"
```

Defines

- #define **STATE_MACHINE_DATA** struct wpa_state_machine
- #define **STATE_MACHINE_DEBUG_PREFIX** "WPA"
- #define **STATE_MACHINE_ADDR** sm->addr
- #define **RSN_SUITE** "%02x-%02x-%02x-%d"
- #define **RSN_SUITE_ARG**(s) ((s) >> 24) & 0xff, ((s) >> 16) & 0xff, ((s) >> 8) & 0xff, (s) & 0xff

Functions

- int **wpa_auth_for_each_sta** (struct wpa_authenticator *wpa_auth, int(*cb)(struct wpa_state_machine *sm, void *ctx), void *cb_ctx)
- int **wpa_auth_for_each_auth** (struct wpa_authenticator *wpa_auth, int(*cb)(struct wpa_authenticator *a, void *ctx), void *cb_ctx)
- void **wpa_auth_logger** (struct wpa_authenticator *wpa_auth, const u8 *addr, logger_level level, const char *txt)
- void **wpa_auth_vlogger** (struct wpa_authenticator *wpa_auth, const u8 *addr, logger_level level, const char *fmt,...)
- struct wpa_authenticator * wpa_init (const u8 *addr, struct wpa_auth_config *conf, struct wpa_auth_callbacks *cb)

  Initialize WPA authenticator.

- void wpa_deinit (struct wpa_authenticator *wpa_auth)

  Deinitialize WPA authenticator.

- int wpa_reconfig (struct wpa_authenticator *wpa_auth, struct wpa_auth_config *conf)

  Update WPA authenticator configuration.

- struct wpa_state_machine * **wpa_auth_sta_init** (struct wpa_authenticator *wpa_auth, const u8 *addr)
- void **wpa_auth_sta_associated** (struct wpa_authenticator *wpa_auth, struct wpa_state_machine *sm)
- void **wpa_auth_sta_no_wpa** (struct wpa_state_machine *sm)
- void **wpa_auth_sta_deinit** (struct wpa_state_machine *sm)
- void **wpa_receive** (struct wpa_authenticator *wpa_auth, struct wpa_state_machine *sm, u8 *data, size_t data_len)
- void **__wpa_send_eapol** (struct wpa_authenticator *wpa_auth, struct wpa_state_machine *sm, int key_info, const u8 *key_rsc, const u8 *nonce, const u8 *kde, size_t kde_len, int keyidx, int encr, int force_version)
- void **wpa_remove_ptk** (struct wpa_state_machine *sm)
- void **wpa_auth_sm_event** (struct wpa_state_machine *sm, wpa_event event)
- **SM_STATE** (WPA_PTK, INITIALIZE)
- **SM_STATE** (WPA_PTK, DISCONNECT)
- **SM_STATE** (WPA_PTK, DISCONNECTED)
- **SM_STATE** (WPA_PTK, AUTHENTICATION)
- **SM_STATE** (WPA_PTK, AUTHENTICATION2)
- **SM_STATE** (WPA_PTK, INITPMK)
- **SM_STATE** (WPA_PTK, INITPSK)
- **SM_STATE** (WPA_PTK, PTKSTART)
- **SM_STATE** (WPA_PTK, PTKCALCNEGOTIATING)
- **SM_STATE** (WPA_PTK, PTKCALCNEGOTIATING2)
- **SM_STATE** (WPA_PTK, PTKINITNEGOTIATING)
- **SM_STATE** (WPA_PTK, PTKINITDONE)
- **SM_STEP** (WPA_PTK)
- **SM_STATE** (WPA_PTK_GROUP, IDLE)
- **SM_STATE** (WPA_PTK_GROUP, REKEYNEGOTIATING)
- **SM_STATE** (WPA_PTK_GROUP, REKEYESTABLISHED)
- **SM_STATE** (WPA_PTK_GROUP, KEYERROR)
- **SM_STEP** (WPA_PTK_GROUP)
- void **wpa_auth_sm_notify** (struct wpa_state_machine *sm)
- void **wpa_gtk_rekey** (struct wpa_authenticator *wpa_auth)
- int **wpa_get_mib** (struct wpa_authenticator *wpa_auth, char *buf, size_t buflen)
- int **wpa_get_mib_sta** (struct wpa_state_machine *sm, char *buf, size_t buflen)
- void **wpa_auth_countermeasures_start** (struct wpa_authenticator *wpa_auth)
- int **wpa_auth_pairwise_set** (struct wpa_state_machine *sm)
- int **wpa_auth_get_pairwise** (struct wpa_state_machine *sm)
- int **wpa_auth_sta_key_mgmt** (struct wpa_state_machine *sm)
- int **wpa_auth_sta_wpa_version** (struct wpa_state_machine *sm)
- int **wpa_auth_sta_clear_pmksa** (struct wpa_state_machine *sm, struct rsn_pmksa_cache_entry *entry)
- struct rsn_pmksa_cache_entry * **wpa_auth_sta_get_pmksa** (struct wpa_state_machine *sm)
- void **wpa_auth_sta_local_mic_failure_report** (struct wpa_state_machine *sm)
- const u8 * **wpa_auth_get_wpa_ie** (struct wpa_authenticator *wpa_auth, size_t *len)
- int **wpa_auth_pmksa_add** (struct wpa_state_machine *sm, const u8 *pmk, int session_timeout, struct eapol_state_machine *eapol)
- int **wpa_auth_pmksa_add_preauth** (struct wpa_authenticator *wpa_auth, const u8 *pmk, size_t len, const u8 *sta_addr, int session_timeout, struct eapol_state_machine *eapol)
- int **wpa_auth_sta_set_vlan** (struct wpa_state_machine *sm, int vlan_id)

15.61.1 Detailed Description

hostapd - IEEE 802.11i-2004 / WPA Authenticator

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.61.2 Function Documentation

15.61.2.1 void wpa_deinit (struct wpa_authenticator * *wpa_auth*)

Deinitialize WPA authenticator.

Parameters:

wpa_auth Pointer to WPA authenticator data from wpa_init()

15.61.2.2 struct wpa_authenticator* wpa_init (const u8 * *addr*, struct wpa_auth_config * *conf*, struct wpa_auth_callbacks * *cb*) [read]

Initialize WPA authenticator.

Parameters:

addr Authenticator address

conf Configuration for WPA authenticator

cb Callback functions for WPA authenticator

Returns:

Pointer to WPA authenticator data or NULL on failure

15.61.2.3 int wpa_reconfig (struct wpa_authenticator * *wpa_auth*, struct wpa_auth_config * *conf*)

Update WPA authenticator configuration.

Parameters:

wpa_auth Pointer to WPA authenticator data from wpa_init()

conf Configuration for WPA authenticator

15.62 src/rsn_supp/wpa.c File Reference

WPA Supplicant - WPA state machine and EAPOL-Key processing. `#include "includes.h"`

```
#include "common.h"
#include "crypto.h"
#include "aes_wrap.h"
#include "wpa.h"
#include "eloop.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_i.h"
#include "wpa_ie.h"
#include "peerkey.h"
#include "ieee802_11_defs.h"
```

Data Structures

- struct wpa_gtk_data

Defines

- #define **RSN_SUITE** "%02x-%02x-%02x-%d"
- #define **RSN_SUITE_ARG**(s) ((s) >> 24) & 0xff, ((s) >> 16) & 0xff, ((s) >> 8) & 0xff, (s) & 0xff

Functions

- void wpa_eapol_key_send (struct wpa_sm *sm, const u8 *kck, int ver, const u8 *dest, u16 proto, u8 *msg, size_t msg_len, u8 *key_mic)

  Send WPA/RSN EAPOL-Key message.

- void wpa_sm_key_request (struct wpa_sm *sm, int error, int pairwise)

  Send EAPOL-Key Request.

- int wpa_supplicant_send_2_of_4 (struct wpa_sm *sm, const unsigned char *dst, const struct wpa_eapol_key *key, int ver, const u8 *nonce, const u8 *wpa_ie, size_t wpa_ie_len, struct wpa_ptk *ptk)

  Send message 2 of WPA/RSN 4-Way Handshake.

- int wpa_supplicant_send_4_of_4 (struct wpa_sm *sm, const unsigned char *dst, const struct wpa_eapol_key *key, u16 ver, u16 key_info, const u8 *kde, size_t kde_len, struct wpa_ptk *ptk)

  Send message 4 of WPA/RSN 4-Way Handshake.

- 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.
- struct 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 rsn_supp_config *config)
  Notification of current configration change.
- void wpa_sm_set_own_addr (struct wpa_sm *sm, const u8 *addr)
  Set own MAC address.
- void wpa_sm_set_ifname (struct wpa_sm *sm, const char *ifname, const char *bridge_ifname)
  Set network interface name.
- void wpa_sm_set_eapol (struct wpa_sm *sm, struct eapol_sm *eapol)
  Set EAPOL state machine pointer.
- int wpa_sm_set_param (struct wpa_sm *sm, enum wpa_sm_conf_params param, unsigned int value)
  Set WPA state machine parameters.
- unsigned int wpa_sm_get_param (struct wpa_sm *sm, enum wpa_sm_conf_params param)

Get WPA state machine parameters.

- int wpa_sm_get_status (struct wpa_sm *sm, char *buf, size_t buflen, int verbose)
  Get WPA state machine.
- int wpa_sm_set_assoc_wpa_ie_default (struct wpa_sm *sm, u8 *wpa_ie, size_t *wpa_ie_len)
  Generate own WPA/RSN IE from configuration.
- int wpa_sm_set_assoc_wpa_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  Set own WPA/RSN IE from (Re)AssocReq.
- int wpa_sm_set_ap_wpa_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  Set AP WPA IE from Beacon/ProbeResp.
- int wpa_sm_set_ap_rsn_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  Set AP RSN IE from Beacon/ProbeResp.
- int wpa_sm_parse_own_wpa_ie (struct wpa_sm *sm, struct wpa_ie_data *data)
  Parse own WPA/RSN IE.
- int **wpa_sm_pmksa_cache_list** (struct wpa_sm *sm, char *buf, size_t len)

15.62.1 Detailed Description

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

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.62.2 Function Documentation

15.62.2.1 void wpa_eapol_key_send (struct wpa_sm * *sm*, const u8 * *kck*, int *ver*, const u8 * *dest*, u16 *proto*, u8 * *msg*, size_t *msg_len*, u8 * *key_mic*)

Send WPA/RSN EAPOL-Key message.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

kck Key Confirmation Key (KCK, part of PTK)

ver Version field from Key Info

dest Destination address for the frame

proto Ethertype (usually ETH_P_EAPOL)

msg EAPOL-Key message

msg_len Length of message

key_mic Pointer to the buffer to which the EAPOL-Key MIC is written

15.62.2.2 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()

15.62.2.3 void wpa_sm_deinit (struct wpa_sm * *sm*)

Deinitialize WPA state machine.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

15.62.2.4 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.

15.62.2.5 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

15.62.2.6 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.

15.62.2.7 struct wpa_sm* wpa_sm_init (struct wpa_sm_ctx * *ctx*) [read]

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.

15.62.2.8 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

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.

15.62.2.9 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.

15.62.2.10 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.

15.62.2.11 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.

15.62.2.12 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.

15.62.2.13 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.

15.62.2.14 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.

15.62.2.15 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().

15.62.2.16 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

15.62.2.17 void wpa_sm_set_config (struct wpa_sm * *sm*, struct rsn_supp_config * *config*)

Notification of current configration change.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

config Pointer to current network configuration

Notify WPA state machine that configuration has changed. config will be stored as a backpointer to network configuration. This can be NULL to clear the stored pointed.

15.62.2.18 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()

15.62.2.19 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

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

Set network interface name.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

ifname Interface name

bridge_ifname Optional bridge interface name (for pre-auth)

15.62.2.21 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

15.62.2.22 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

15.62.2.23 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.

15.62.2.24 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.

15.62.2.25 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

15.62.2.26 int wpa_supplicant_send_2_of_4 (struct wpa_sm * *sm*, const unsigned char * *dst*, const struct wpa_eapol_key * *key*, int *ver*, const u8 * *nonce*, const u8 * *wpa_ie*, size_t *wpa_ie_len*, struct wpa_ptk * *ptk*)

Send message 2 of WPA/RSN 4-Way Handshake.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

dst Destination address for the frame

key Pointer to the EAPOL-Key frame header

ver Version bits from EAPOL-Key Key Info

nonce Nonce value for the EAPOL-Key frame

wpa_ie WPA/RSN IE

wpa_ie_len Length of the WPA/RSN IE

ptk PTK to use for keyed hash and encryption

Returns:

0 on success, -1 on failure

15.62.2.27 int wpa_supplicant_send_4_of_4 (struct wpa_sm * *sm*, const unsigned char * *dst*, const struct wpa_eapol_key * *key*, u16 *ver*, u16 *key_info*, const u8 * *kde*, size_t *kde_len*, struct wpa_ptk * *ptk*)

Send message 4 of WPA/RSN 4-Way Handshake.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

dst Destination address for the frame

key Pointer to the EAPOL-Key frame header

ver Version bits from EAPOL-Key Key Info

key_info Key Info

kde KDEs to include the EAPOL-Key frame

kde_len Length of KDEs

ptk PTK to use for keyed hash and encryption

Returns:

0 on success, -1 on failure

15.63 hostapd/wpa.h File Reference

hostapd - IEEE 802.11i-2004 / WPA Authenticator `#include "defs.h"`

`#include "eapol_common.h"`

`#include "wpa_common.h"`

Data Structures

- struct ft_rrb_frame
- struct ft_r0kh_r1kh_pull_frame
- struct ft_r0kh_r1kh_resp_frame
- struct ft_r0kh_r1kh_push_frame
- struct ft_remote_r0kh
- struct ft_remote_r1kh
- struct wpa_auth_config
- struct wpa_auth_callbacks

Defines

- #define **RSN_REMOTE_FRAME_TYPE_FT_RRB** 1
- #define **FT_PACKET_REQUEST** 0
- #define **FT_PACKET_RESPONSE** 1
- #define **FT_PACKET_R0KH_R1KH_PULL** 200
- #define **FT_PACKET_R0KH_R1KH_RESP** 201
- #define **FT_PACKET_R0KH_R1KH_PUSH** 202
- #define **FT_R0KH_R1KH_PULL_DATA_LEN** 44
- #define **FT_R0KH_R1KH_RESP_DATA_LEN** 76
- #define **FT_R0KH_R1KH_PUSH_DATA_LEN** 80

Enumerations

- enum **logger_level** { **LOGGER_DEBUG**, **LOGGER_INFO**, **LOGGER_WARNING** }
- enum **wpa_eapol_variable** {

  WPA_EAPOL_portEnabled, **WPA_EAPOL_portValid**, **WPA_EAPOL_authorized**, **WPA_-EAPOL_portControl_Auto**,

  WPA_EAPOL_keyRun, **WPA_EAPOL_keyAvailable**, **WPA_EAPOL_keyDone**, **WPA_-EAPOL_inc_EapolFramesTx** }
- enum {

  WPA_IE_OK, **WPA_INVALID_IE**, **WPA_INVALID_GROUP**, **WPA_INVALID_PAIRWISE**,

  WPA_INVALID_AKMP, **WPA_NOT_ENABLED**, **WPA_ALLOC_FAIL**, **WPA_MGMT_-FRAME_PROTECTION_VIOLATION**,

  WPA_INVALID_MGMT_GROUP_CIPHER, **WPA_INVALID_MDIE**, **WPA_INVALID_-PROTO** }
- enum **wpa_event** {

  WPA_AUTH, **WPA_ASSOC**, **WPA_DISASSOC**, **WPA_DEAUTH**,

  WPA_REAUTH, **WPA_REAUTH_EAPOL**, **WPA_ASSOC_FT** }

Functions

- struct wpa_authenticator * wpa_init (const u8 *addr, struct wpa_auth_config *conf, struct wpa_auth_callbacks *cb)

  Initialize WPA authenticator.

- void wpa_deinit (struct wpa_authenticator *wpa_auth)

  Deinitialize WPA authenticator.

- int wpa_reconfig (struct wpa_authenticator *wpa_auth, struct wpa_auth_config *conf)

  Update WPA authenticator configuration.

- int **wpa_validate_wpa_ie** (struct wpa_authenticator *wpa_auth, struct wpa_state_machine *sm, const u8 *wpa_ie, size_t wpa_ie_len, const u8 *mdie, size_t mdie_len)
- int **wpa_auth_uses_mfp** (struct wpa_state_machine *sm)
- struct wpa_state_machine * **wpa_auth_sta_init** (struct wpa_authenticator *wpa_auth, const u8 *addr)
- void **wpa_auth_sta_associated** (struct wpa_authenticator *wpa_auth, struct wpa_state_machine *sm)
- void **wpa_auth_sta_no_wpa** (struct wpa_state_machine *sm)
- void **wpa_auth_sta_deinit** (struct wpa_state_machine *sm)
- void **wpa_receive** (struct wpa_authenticator *wpa_auth, struct wpa_state_machine *sm, u8 *data, size_t data_len)
- void **wpa_remove_ptk** (struct wpa_state_machine *sm)
- void **wpa_auth_sm_event** (struct wpa_state_machine *sm, wpa_event event)
- void **wpa_auth_sm_notify** (struct wpa_state_machine *sm)
- void **wpa_gtk_rekey** (struct wpa_authenticator *wpa_auth)
- int **wpa_get_mib** (struct wpa_authenticator *wpa_auth, char *buf, size_t buflen)
- int **wpa_get_mib_sta** (struct wpa_state_machine *sm, char *buf, size_t buflen)
- void **wpa_auth_countermeasures_start** (struct wpa_authenticator *wpa_auth)
- int **wpa_auth_pairwise_set** (struct wpa_state_machine *sm)
- int **wpa_auth_get_pairwise** (struct wpa_state_machine *sm)
- int **wpa_auth_sta_key_mgmt** (struct wpa_state_machine *sm)
- int **wpa_auth_sta_wpa_version** (struct wpa_state_machine *sm)
- int **wpa_auth_sta_clear_pmksa** (struct wpa_state_machine *sm, struct rsn_pmksa_cache_entry *entry)
- struct rsn_pmksa_cache_entry * **wpa_auth_sta_get_pmksa** (struct wpa_state_machine *sm)
- void **wpa_auth_sta_local_mic_failure_report** (struct wpa_state_machine *sm)
- const u8 * **wpa_auth_get_wpa_ie** (struct wpa_authenticator *wpa_auth, size_t *len)
- int **wpa_auth_pmksa_add** (struct wpa_state_machine *sm, const u8 *pmk, int session_timeout, struct eapol_state_machine *eapol)
- int **wpa_auth_pmksa_add_preauth** (struct wpa_authenticator *wpa_auth, const u8 *pmk, size_t len, const u8 *sta_addr, int session_timeout, struct eapol_state_machine *eapol)
- int **wpa_auth_sta_set_vlan** (struct wpa_state_machine *sm, int vlan_id)

Variables

- struct ft_rrb_frame **STRUCT_PACKED**

15.63.1 Detailed Description

hostapd - IEEE 802.11i-2004 / WPA Authenticator

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.63.2 Function Documentation

15.63.2.1 void wpa_deinit (struct wpa_authenticator * *wpa_auth*)

Deinitialize WPA authenticator.

Parameters:

wpa_auth Pointer to WPA authenticator data from wpa_init()

15.63.2.2 struct wpa_authenticator* wpa_init (const u8 * *addr*, struct wpa_auth_config * *conf*, struct wpa_auth_callbacks * *cb*) [read]

Initialize WPA authenticator.

Parameters:

addr Authenticator address

conf Configuration for WPA authenticator

cb Callback functions for WPA authenticator

Returns:

Pointer to WPA authenticator data or NULL on failure

15.63.2.3 int wpa_reconfig (struct wpa_authenticator * *wpa_auth*, struct wpa_auth_config * *conf*)

Update WPA authenticator configuration.

Parameters:

wpa_auth Pointer to WPA authenticator data from wpa_init()

conf Configuration for WPA authenticator

15.64 src/rsn_supp/wpa.h File Reference

wpa_supplicant - WPA definitions `#include "defs.h"`

`#include "eapol_common.h"`

`#include "wpa_common.h"`

Data Structures

- struct wpa_sm_ctx
- struct rsn_supp_config

Enumerations

- enum **wpa_sm_conf_params** {

  RSNA_PMK_LIFETIME, **RSNA_PMK_REAUTH_THRESHOLD**, **RSNA_SA_TIMEOUT**, **WPA_PARAM_PROTO**,

  WPA_PARAM_PAIRWISE, **WPA_PARAM_GROUP**, **WPA_PARAM_KEY_MGMT**, **WPA_PARAM_MGMT_GROUP**,

  WPA_PARAM_RSN_ENABLED }

Functions

- struct 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 rsn_supp_config *config)

Notification of current configration change.

- void wpa_sm_set_own_addr (struct wpa_sm *sm, const u8 *addr)
  Set own MAC address.
- void wpa_sm_set_ifname (struct wpa_sm *sm, const char *ifname, const char *bridge_ifname)
  Set network interface name.
- void wpa_sm_set_eapol (struct wpa_sm *sm, struct eapol_sm *eapol)
  Set EAPOL state machine pointer.
- int wpa_sm_set_assoc_wpa_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  Set own WPA/RSN IE from (Re)AssocReq.
- int wpa_sm_set_assoc_wpa_ie_default (struct wpa_sm *sm, u8 *wpa_ie, size_t *wpa_ie_len)
  Generate own WPA/RSN IE from configuration.
- int wpa_sm_set_ap_wpa_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  Set AP WPA IE from Beacon/ProbeResp.
- int wpa_sm_set_ap_rsn_ie (struct wpa_sm *sm, const u8 *ie, size_t len)
  Set AP RSN IE from Beacon/ProbeResp.
- int wpa_sm_get_mib (struct wpa_sm *sm, char *buf, size_t buflen)
  Dump text list of MIB entries.
- int wpa_sm_set_param (struct wpa_sm *sm, enum wpa_sm_conf_params param, unsigned int value)
  Set WPA state machine parameters.
- unsigned int wpa_sm_get_param (struct wpa_sm *sm, enum wpa_sm_conf_params param)
  Get WPA state machine parameters.
- int wpa_sm_get_status (struct wpa_sm *sm, char *buf, size_t buflen, int verbose)
  Get WPA state machine.
- void wpa_sm_key_request (struct wpa_sm *sm, int error, int pairwise)
  Send EAPOL-Key Request.
- int wpa_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.
- int **wpa_sm_pmksa_cache_list** (struct wpa_sm *sm, char *buf, size_t len)

15.64.1 Detailed Description

wpa_supplicant - WPA definitions

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.64.2 Function Documentation

15.64.2.1 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.

15.64.2.2 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()

15.64.2.3 void wpa_sm_deinit (struct wpa_sm * *sm*)

Deinitialize WPA state machine.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

15.64.2.4 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.

15.64.2.5 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

15.64.2.6 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.

15.64.2.7 struct wpa_sm* wpa_sm_init (struct wpa_sm_ctx * *ctx*) [read]

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.

15.64.2.8 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

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.

15.64.2.9 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.

15.64.2.10 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.

15.64.2.11 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.

15.64.2.12 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.

15.64.2.13 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.

15.64.2.14 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.

15.64.2.15 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().

15.64.2.16 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

15.64.2.17 void wpa_sm_set_config (struct wpa_sm * *sm*, struct rsn_supp_config * *config*)

Notification of current configration change.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

config Pointer to current network configuration

Notify WPA state machine that configuration has changed. config will be stored as a backpointer to network configuration. This can be NULL to clear the stored pointed.

15.64.2.18 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()

15.64.2.19 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

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

Set network interface name.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

ifname Interface name

bridge_ifname Optional bridge interface name (for pre-auth)

15.64.2.21 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

15.64.2.22 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

15.64.2.23 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.

15.64.2.24 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.

15.64.2.25 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

15.65 hostapd/wpa_auth_i.h File Reference

hostapd - IEEE 802.11i-2004 / WPA Authenticator: Internal definitions

Data Structures

- struct wpa_stsl_negotiation
- struct wpa_state_machine
- struct wpa_group
- struct wpa_authenticator

Defines

- #define **RSNA_MAX_EAPOL_RETRIES** 4

Functions

- int **wpa_write_rsn_ie** (struct wpa_auth_config *conf, u8 *buf, size_t len, const u8 *pmkid)
- void **wpa_auth_logger** (struct wpa_authenticator *wpa_auth, const u8 *addr, logger_level level, const char *txt)
- void **wpa_auth_vlogger** (struct wpa_authenticator *wpa_auth, const u8 *addr, logger_level level, const char *fmt,...)
- void **__wpa_send_eapol** (struct wpa_authenticator *wpa_auth, struct wpa_state_machine *sm, int key_info, const u8 *key_rsc, const u8 *nonce, const u8 *kde, size_t kde_len, int keyidx, int encr, int force_version)
- int **wpa_auth_for_each_sta** (struct wpa_authenticator *wpa_auth, int(*cb)(struct wpa_state_machine *sm, void *ctx), void *cb_ctx)
- int **wpa_auth_for_each_auth** (struct wpa_authenticator *wpa_auth, int(*cb)(struct wpa_authenticator *a, void *ctx), void *cb_ctx)

15.65.1 Detailed Description

hostapd - IEEE 802.11i-2004 / WPA Authenticator: Internal definitions

Copyright

15.66 hostapd/wpa_auth_ie.c File Reference

hostapd - WPA/RSN IE and KDE definitions `#include "includes.h"`

```
#include "common.h"
#include "config.h"
#include "ieee802_11.h"
#include "eapol_sm.h"
#include "wpa.h"
#include "pmksa_cache.h"
#include "wpa_auth_ie.h"
#include "wpa_auth_i.h"
```

Data Structures

- struct wpa_auth_okc_iter_data

Functions

- int **wpa_write_rsn_ie** (struct wpa_auth_config *conf, u8 *buf, size_t len, const u8 *pmkid)
- int **wpa_auth_gen_wpa_ie** (struct wpa_authenticator *wpa_auth)
- u8 * **wpa_add_kde** (u8 *pos, u32 kde, const u8 *data, size_t data_len, const u8 *data2, size_t data2_len)
- int **wpa_validate_wpa_ie** (struct wpa_authenticator *wpa_auth, struct wpa_state_machine *sm, const u8 *wpa_ie, size_t wpa_ie_len, const u8 *mdie, size_t mdie_len)
- int wpa_parse_kde_ies (const u8 *buf, size_t len, struct wpa_eapol_ie_parse *ie)

  Parse EAPOL-Key Key Data IEs.

- int **wpa_auth_uses_mfp** (struct wpa_state_machine *sm)

15.66.1 Detailed Description

hostapd - WPA/RSN IE and KDE definitions

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.66.2 Function Documentation

15.66.2.1 int wpa_parse_kde_ies (const u8 * *buf*, size_t *len*, struct wpa_eapol_ie_parse * *ie*)

Parse EAPOL-Key Key Data IEs.

Parameters:

buf Pointer to the Key Data buffer

len Key Data Length

ie Pointer to parsed IE data

Returns:

0 on success, -1 on failure

15.67 hostapd/wpa_auth_ie.h File Reference

hostapd - WPA/RSN IE and KDE definitions

Data Structures

- struct wpa_eapol_ie_parse

Functions

- int wpa_parse_kde_ies (const u8 *buf, size_t len, struct wpa_eapol_ie_parse *ie)

  Parse EAPOL-Key Key Data IEs.

- u8 * **wpa_add_kde** (u8 *pos, u32 kde, const u8 *data, size_t data_len, const u8 *data2, size_t data2_len)
- int **wpa_auth_gen_wpa_ie** (struct wpa_authenticator *wpa_auth)

15.67.1 Detailed Description

hostapd - WPA/RSN IE and KDE definitions

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.67.2 Function Documentation

15.67.2.1 int wpa_parse_kde_ies (const u8 * *buf*, size_t *len*, struct wpa_eapol_ie_parse * *ie*)

Parse EAPOL-Key Key Data IEs.

Parameters:

buf Pointer to the Key Data buffer

len Key Data Length

ie Pointer to parsed IE data

Returns:

0 on success, -1 on failure

15.68 hostapd/wpa_ft.c File Reference

hostapd - IEEE 802.11r - Fast BSS Transition `#include "includes.h"`

```
#include "common.h"
#include "config.h"
#include "wpa.h"
#include "aes_wrap.h"
#include "ieee802_11.h"
#include "wme.h"
#include "defs.h"
#include "wpa_auth_i.h"
#include "wpa_auth_ie.h"
```

15.68.1 Detailed Description

hostapd - IEEE 802.11r - Fast BSS Transition

Copyright

15.69 src/rsn_supp/wpa_ft.c File Reference

WPA Supplicant - IEEE 802.11r - Fast BSS Transition. `#include "includes.h"`

```
#include "common.h"
#include "wpa.h"
#include "wpa_i.h"
#include "wpa_ie.h"
#include "aes_wrap.h"
#include "ieee802_11_defs.h"
#include "ieee802_11_common.h"
```

15.69.1 Detailed Description

WPA Supplicant - IEEE 802.11r - Fast BSS Transition.

Copyright

15.70 hostapd/wps_hostapd.c File Reference

hostapd / WPS integration `#include "includes.h"`

```
#include "hostapd.h"
#include "driver_i.h"
#include "eloop.h"
#include "uuid.h"
#include "wpa_ctrl.h"
#include "ieee802_11_defs.h"
#include "ieee802_11_common.h"
#include "sta_info.h"
#include "eapol_sm.h"
#include "wps/wps.h"
#include "wps/wps_defs.h"
#include "wps/wps_dev_attr.h"
#include "wps_hostapd.h"
#include "dh_groups.h"
```

Functions

- int **hostapd_init_wps** (struct hostapd_data *hapd, struct hostapd_bss_config *conf)
- void **hostapd_deinit_wps** (struct hostapd_data *hapd)
- int **hostapd_wps_add_pin** (struct hostapd_data *hapd, const char *uuid, const char *pin, int timeout)
- int **hostapd_wps_button_pushed** (struct hostapd_data *hapd)
- int **hostapd_wps_get_mib_sta** (struct hostapd_data *hapd, const u8 *addr, char *buf, size_t buflen)

15.70.1 Detailed Description

hostapd / WPS integration

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.71 hostapd/wps_hostapd.h File Reference

hostapd / WPS integration

15.71.1 Detailed Description

hostapd / WPS integration

Copyright

15.72 src/common/defs.h File Reference

WPA Supplicant - Common definitions.

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_KEY_MGMT_FT_IEEE8021X** BIT(5)
- #define **WPA_KEY_MGMT_FT_PSK** BIT(6)
- #define **WPA_KEY_MGMT_IEEE8021X_SHA256** BIT(7)
- #define **WPA_KEY_MGMT_PSK_SHA256** BIT(8)
- #define **WPA_KEY_MGMT_WPS** BIT(9)
- #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 **MLME_SETPROTECTION_PROTECT_TYPE_NONE** 0
- #define **MLME_SETPROTECTION_PROTECT_TYPE_RX** 1
- #define **MLME_SETPROTECTION_PROTECT_TYPE_TX** 2
- #define **MLME_SETPROTECTION_PROTECT_TYPE_RX_TX** 3
- #define **MLME_SETPROTECTION_KEY_TYPE_GROUP** 0
- #define **MLME_SETPROTECTION_KEY_TYPE_PAIRWISE** 1

Enumerations

- enum **Boolean** { **FALSE** = 0, **TRUE** = 1 }
- enum **wpa_alg** {

  WPA_ALG_NONE, **WPA_ALG_WEP**, **WPA_ALG_TKIP**, **WPA_ALG_CCMP**,

  WPA_ALG_IGTK, **WPA_ALG_PMK** }
- 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, **KEY_MGMT_FT_802_1X**, **KEY_MGMT_FT_PSK**, **KEY_-MGMT_802_1X_SHA256**,

  KEY_MGMT_PSK_SHA256, **KEY_MGMT_WPS** }

- enum wpa_states {

  WPA_DISCONNECTED, WPA_INACTIVE, WPA_SCANNING, WPA_AUTHENTICATING,

  WPA_ASSOCIATING, WPA_ASSOCIATED, WPA_4WAY_HANDSHAKE, WPA_GROUP_-HANDSHAKE,

  WPA_COMPLETED }
- enum **mfp_options** { **NO_IEEE80211W** = 0, **IEEE80211W_OPTIONAL** = 1, **IEEE80211W_-REQUIRED** = 2 }
- enum **hostapd_hw_mode** { **HOSTAPD_MODE_IEEE80211B**, **HOSTAPD_MODE_-IEEE80211G**, **HOSTAPD_MODE_IEEE80211A**, **NUM_HOSTAPD_MODES** }

15.72.1 Detailed Description

WPA Supplicant - Common definitions.

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.72.2 Enumeration Type Documentation

15.72.2.1 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.

Enumerator:

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_AUTHENTICATING Trying to authenticate with a BSS/SSID. This state is entered when wpa_supplicant has found a suitable BSS to authenticate with and the driver is configured to try to authenticate with this BSS. This state is used only with drivers that use wpa_supplicant as the SME.

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.

15.73 src/common/eapol_common.h File Reference

EAPOL definitions shared between hostapd and wpa_supplicant.

Data Structures

- struct ieee802_1x_hdr

Defines

- #define **EAPOL_VERSION** 2

Enumerations

- enum {
  IEEE802_1X_TYPE_EAP_PACKET = 0, **IEEE802_1X_TYPE_EAPOL_START** = 1, **IEEE802_1X_TYPE_EAPOL_LOGOFF** = 2, **IEEE802_1X_TYPE_EAPOL_KEY** = 3, **IEEE802_1X_TYPE_EAPOL_ENCAPSULATED_ASF_ALERT** = 4 }
- enum { **EAPOL_KEY_TYPE_RC4** = 1, **EAPOL_KEY_TYPE_RSN** = 2, **EAPOL_KEY_TYPE_WPA** = 254 }

Variables

- struct ieee802_1x_hdr **STRUCT_PACKED**

15.73.1 Detailed Description

EAPOL definitions shared between hostapd and wpa_supplicant.

Copyright

15.74 src/common/ieee802_11_common.c File Reference

IEEE 802.11 Common routines. `#include "includes.h"`

`#include "common.h"`

`#include "ieee802_11_defs.h"`

`#include "ieee802_11_common.h"`

Functions

- ParseRes ieee802_11_parse_elems (u8 *start, size_t len, struct ieee802_11_elems *elems, int show_errors)

  Parse information elements in management frames.

- int **ieee802_11_ie_count** (const u8 *ies, size_t ies_len)
- struct wpabuf * **ieee802_11_vendor_ie_concat** (const u8 *ies, size_t ies_len, u32 oui_type)

15.74.1 Detailed Description

IEEE 802.11 Common routines.

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.74.2 Function Documentation

15.74.2.1 ParseRes ieee802_11_parse_elems (u8 * *start*, size_t *len*, struct ieee802_11_elems * *elems*, int *show_errors*)

Parse information elements in management frames.

Parameters:

start Pointer to the start of IEs

len Length of IE buffer in octets

elems Data structure for parsed elements

show_errors Whether to show parsing errors in debug log

Returns:

Parsing result

15.75 src/common/ieee802_11_common.h File Reference

IEEE 802.11 Common routines.

Data Structures

- struct ieee802_11_elems

Enumerations

- enum **ParseRes** { **ParseOK** = 0, **ParseUnknown** = 1, **ParseFailed** = -1 }

Functions

- ParseRes ieee802_11_parse_elems (u8 *start, size_t len, struct ieee802_11_elems *elems, int show_errors)

  Parse information elements in management frames.

- int **ieee802_11_ie_count** (const u8 *ies, size_t ies_len)
- struct wpabuf * **ieee802_11_vendor_ie_concat** (const u8 *ies, size_t ies_len, u32 oui_type)

15.75.1 Detailed Description

IEEE 802.11 Common routines.

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.75.2 Function Documentation

15.75.2.1 ParseRes ieee802_11_parse_elems (u8 * *start*, size_t *len*, struct ieee802_11_elems * *elems*, int *show_errors*)

Parse information elements in management frames.

Parameters:

start Pointer to the start of IEs

len Length of IE buffer in octets

elems Data structure for parsed elements

show_errors Whether to show parsing errors in debug log

Returns:

Parsing result

15.76 src/common/ieee802_11_defs.h File Reference

IEEE 802.11 Frame type definitions.

Data Structures

- struct ieee80211_hdr
- struct ieee80211_mgmt
- struct ieee80211_ht_capability
- struct ieee80211_ht_operation
- struct ht_cap_ie
- struct secondary_channel_offset_ie
- struct recommended_tx_channel_width_action
- struct mimo_pwr_save_action
- struct wmm_information_element
- struct wmm_ac_parameter
- struct wmm_parameter_element
- struct wmm_tspec_element

Defines

- #define **WLAN_FC_PVER** 0x0003
- #define **WLAN_FC_TODS** 0x0100
- #define **WLAN_FC_FROMDS** 0x0200
- #define **WLAN_FC_MOREFRAG** 0x0400
- #define **WLAN_FC_RETRY** 0x0800
- #define **WLAN_FC_PWRMGT** 0x1000
- #define **WLAN_FC_MOREDATA** 0x2000
- #define **WLAN_FC_ISWEP** 0x4000
- #define **WLAN_FC_ORDER** 0x8000
- #define **WLAN_FC_GET_TYPE**(fc) (((fc) & 0x000c) >> 2)
- #define **WLAN_FC_GET_STYPE**(fc) (((fc) & 0x00f0) >> 4)
- #define **WLAN_GET_SEQ_FRAG**(seq) ((seq) & (BIT(3) | BIT(2) | BIT(1) | BIT(0)))
- #define **WLAN_GET_SEQ_SEQ**(seq) (((seq) & (~(BIT(3) | BIT(2) | BIT(1) | BIT(0)))) >> 4)
- #define **WLAN_FC_TYPE_MGMT** 0
- #define **WLAN_FC_TYPE_CTRL** 1
- #define **WLAN_FC_TYPE_DATA** 2
- #define **WLAN_FC_STYPE_ASSOC_REQ** 0
- #define **WLAN_FC_STYPE_ASSOC_RESP** 1
- #define **WLAN_FC_STYPE_REASSOC_REQ** 2
- #define **WLAN_FC_STYPE_REASSOC_RESP** 3
- #define **WLAN_FC_STYPE_PROBE_REQ** 4
- #define **WLAN_FC_STYPE_PROBE_RESP** 5
- #define **WLAN_FC_STYPE_BEACON** 8
- #define **WLAN_FC_STYPE_ATIM** 9
- #define **WLAN_FC_STYPE_DISASSOC** 10
- #define **WLAN_FC_STYPE_AUTH** 11
- #define **WLAN_FC_STYPE_DEAUTH** 12
- #define **WLAN_FC_STYPE_ACTION** 13

- #define **WLAN_FC_STYPE_PSPOLL** 10
- #define **WLAN_FC_STYPE_RTS** 11
- #define **WLAN_FC_STYPE_CTS** 12
- #define **WLAN_FC_STYPE_ACK** 13
- #define **WLAN_FC_STYPE_CFEND** 14
- #define **WLAN_FC_STYPE_CFENDACK** 15
- #define **WLAN_FC_STYPE_DATA** 0
- #define **WLAN_FC_STYPE_DATA_CFACK** 1
- #define **WLAN_FC_STYPE_DATA_CFPOLL** 2
- #define **WLAN_FC_STYPE_DATA_CFACKPOLL** 3
- #define **WLAN_FC_STYPE_NULLFUNC** 4
- #define **WLAN_FC_STYPE_CFACK** 5
- #define **WLAN_FC_STYPE_CFPOLL** 6
- #define **WLAN_FC_STYPE_CFACKPOLL** 7
- #define **WLAN_FC_STYPE_QOS_DATA** 8
- #define **WLAN_AUTH_OPEN** 0
- #define **WLAN_AUTH_SHARED_KEY** 1
- #define **WLAN_AUTH_FT** 2
- #define **WLAN_AUTH_LEAP** 128
- #define **WLAN_AUTH_CHALLENGE_LEN** 128
- #define **WLAN_CAPABILITY_ESS** BIT(0)
- #define **WLAN_CAPABILITY_IBSS** BIT(1)
- #define **WLAN_CAPABILITY_CF_POLLABLE** BIT(2)
- #define **WLAN_CAPABILITY_CF_POLL_REQUEST** BIT(3)
- #define **WLAN_CAPABILITY_PRIVACY** BIT(4)
- #define **WLAN_CAPABILITY_SHORT_PREAMBLE** BIT(5)
- #define **WLAN_CAPABILITY_PBCC** BIT(6)
- #define **WLAN_CAPABILITY_CHANNEL_AGILITY** BIT(7)
- #define **WLAN_CAPABILITY_SPECTRUM_MGMT** BIT(8)
- #define **WLAN_CAPABILITY_SHORT_SLOT_TIME** BIT(10)
- #define **WLAN_CAPABILITY_DSSS_OFDM** BIT(13)
- #define **WLAN_STATUS_SUCCESS** 0
- #define **WLAN_STATUS_UNSPECIFIED_FAILURE** 1
- #define **WLAN_STATUS_CAPS_UNSUPPORTED** 10
- #define **WLAN_STATUS_REASSOC_NO_ASSOC** 11
- #define **WLAN_STATUS_ASSOC_DENIED_UNSPEC** 12
- #define **WLAN_STATUS_NOT_SUPPORTED_AUTH_ALG** 13
- #define **WLAN_STATUS_UNKNOWN_AUTH_TRANSACTION** 14
- #define **WLAN_STATUS_CHALLENGE_FAIL** 15
- #define **WLAN_STATUS_AUTH_TIMEOUT** 16
- #define **WLAN_STATUS_AP_UNABLE_TO_HANDLE_NEW_STA** 17
- #define **WLAN_STATUS_ASSOC_DENIED_RATES** 18
- #define **WLAN_STATUS_ASSOC_DENIED_NOSHORT** 19
- #define **WLAN_STATUS_ASSOC_DENIED_NOPBCC** 20
- #define **WLAN_STATUS_ASSOC_DENIED_NOAGILITY** 21
- #define **WLAN_STATUS_SPEC_MGMT_REQUIRED** 22
- #define **WLAN_STATUS_PWR_CAPABILITY_NOT_VALID** 23
- #define **WLAN_STATUS_SUPPORTED_CHANNEL_NOT_VALID** 24
- #define **WLAN_STATUS_ASSOC_DENIED_NO_SHORT_SLOT_TIME** 25
- #define **WLAN_STATUS_ASSOC_DENIED_NO_ER_PBCC** 26

- #define **WLAN_STATUS_ASSOC_DENIED_NO_DSSS_OFDM** 27
- #define **WLAN_STATUS_R0KH_UNREACHABLE** 28
- #define **WLAN_STATUS_ASSOC_REJECTED_TEMPORARILY** 30
- #define **WLAN_STATUS_ROBUST_MGMT_FRAME_POLICY_VIOLATION** 31
- #define **WLAN_STATUS_UNSPECIFIED_QOS_FAILURE** 32
- #define **WLAN_STATUS_REQUEST_DECLINED** 37
- #define **WLAN_STATUS_INVALID_PARAMETERS** 38
- #define **WLAN_STATUS_INVALID_IE** 40
- #define **WLAN_STATUS_GROUP_CIPHER_NOT_VALID** 41
- #define **WLAN_STATUS_PAIRWISE_CIPHER_NOT_VALID** 42
- #define **WLAN_STATUS_AKMP_NOT_VALID** 43
- #define **WLAN_STATUS_UNSUPPORTED_RSN_IE_VERSION** 44
- #define **WLAN_STATUS_INVALID_RSN_IE_CAPAB** 45
- #define **WLAN_STATUS_CIPHER_REJECTED_PER_POLICY** 46
- #define **WLAN_STATUS_TS_NOT_CREATED** 47
- #define **WLAN_STATUS_DIRECT_LINK_NOT_ALLOWED** 48
- #define **WLAN_STATUS_DEST_STA_NOT_PRESENT** 49
- #define **WLAN_STATUS_DEST_STA_NOT_QOS_STA** 50
- #define **WLAN_STATUS_ASSOC_DENIED_LISTEN_INT_TOO_LARGE** 51
- #define **WLAN_STATUS_INVALID_FT_ACTION_FRAME_COUNT** 52
- #define **WLAN_STATUS_INVALID_PMKID** 53
- #define **WLAN_STATUS_INVALID_MDIE** 54
- #define **WLAN_STATUS_INVALID_FTIE** 55
- #define **WLAN_REASON_UNSPECIFIED** 1
- #define **WLAN_REASON_PREV_AUTH_NOT_VALID** 2
- #define **WLAN_REASON_DEAUTH_LEAVING** 3
- #define **WLAN_REASON_DISASSOC_DUE_TO_INACTIVITY** 4
- #define **WLAN_REASON_DISASSOC_AP_BUSY** 5
- #define **WLAN_REASON_CLASS2_FRAME_FROM_NONAUTH_STA** 6
- #define **WLAN_REASON_CLASS3_FRAME_FROM_NONASSOC_STA** 7
- #define **WLAN_REASON_DISASSOC_STA_HAS_LEFT** 8
- #define **WLAN_REASON_STA_REQ_ASSOC_WITHOUT_AUTH** 9
- #define **WLAN_REASON_PWR_CAPABILITY_NOT_VALID** 10
- #define **WLAN_REASON_SUPPORTED_CHANNEL_NOT_VALID** 11
- #define **WLAN_REASON_INVALID_IE** 13
- #define **WLAN_REASON_MICHAEL_MIC_FAILURE** 14
- #define **WLAN_REASON_4WAY_HANDSHAKE_TIMEOUT** 15
- #define **WLAN_REASON_GROUP_KEY_UPDATE_TIMEOUT** 16
- #define **WLAN_REASON_IE_IN_4WAY_DIFFERS** 17
- #define **WLAN_REASON_GROUP_CIPHER_NOT_VALID** 18
- #define **WLAN_REASON_PAIRWISE_CIPHER_NOT_VALID** 19
- #define **WLAN_REASON_AKMP_NOT_VALID** 20
- #define **WLAN_REASON_UNSUPPORTED_RSN_IE_VERSION** 21
- #define **WLAN_REASON_INVALID_RSN_IE_CAPAB** 22
- #define **WLAN_REASON_IEEE_802_1X_AUTH_FAILED** 23
- #define **WLAN_REASON_CIPHER_SUITE_REJECTED** 24
- #define **WLAN_EID_SSID** 0
- #define **WLAN_EID_SUPP_RATES** 1
- #define **WLAN_EID_FH_PARAMS** 2
- #define **WLAN_EID_DS_PARAMS** 3

- #define **WLAN_EID_CF_PARAMS** 4
- #define **WLAN_EID_TIM** 5
- #define **WLAN_EID_IBSS_PARAMS** 6
- #define **WLAN_EID_COUNTRY** 7
- #define **WLAN_EID_CHALLENGE** 16
- #define **WLAN_EID_PWR_CONSTRAINT** 32
- #define **WLAN_EID_PWR_CAPABILITY** 33
- #define **WLAN_EID_TPC_REQUEST** 34
- #define **WLAN_EID_TPC_REPORT** 35
- #define **WLAN_EID_SUPPORTED_CHANNELS** 36
- #define **WLAN_EID_CHANNEL_SWITCH** 37
- #define **WLAN_EID_MEASURE_REQUEST** 38
- #define **WLAN_EID_MEASURE_REPORT** 39
- #define **WLAN_EID_QUITE** 40
- #define **WLAN_EID_IBSS_DFS** 41
- #define **WLAN_EID_ERP_INFO** 42
- #define **WLAN_EID_HT_CAP** 45
- #define **WLAN_EID_RSN** 48
- #define **WLAN_EID_EXT_SUPP_RATES** 50
- #define **WLAN_EID_MOBILITY_DOMAIN** 54
- #define **WLAN_EID_FAST_BSS_TRANSITION** 55
- #define **WLAN_EID_TIMEOUT_INTERVAL** 56
- #define **WLAN_EID_RIC_DATA** 57
- #define **WLAN_EID_HT_OPERATION** 61
- #define **WLAN_EID_SECONDARY_CHANNEL_OFFSET** 62
- #define **WLAN_EID_20_40_BSS_COEXISTENCE** 72
- #define **WLAN_EID_20_40_BSS_INTOLERANT** 73
- #define **WLAN_EID_OVERLAPPING_BSS_SCAN_PARAMS** 74
- #define **WLAN_EID_MMIE** 76
- #define **WLAN_EID_VENDOR_SPECIFIC** 221
- #define **WLAN_ACTION_SPECTRUM_MGMT** 0
- #define **WLAN_ACTION_QOS** 1
- #define **WLAN_ACTION_DLS** 2
- #define **WLAN_ACTION_BLOCK_ACK** 3
- #define **WLAN_ACTION_PUBLIC** 4
- #define **WLAN_ACTION_RADIO_MEASUREMENT** 5
- #define **WLAN_ACTION_FT** 6
- #define **WLAN_ACTION_HT** 7
- #define **WLAN_ACTION_SA_QUERY** 8
- #define **WLAN_ACTION_WMM** 17
- #define **WLAN_SA_QUERY_REQUEST** 0
- #define **WLAN_SA_QUERY_RESPONSE** 1
- #define **WLAN_SA_QUERY_TR_ID_LEN** 2
- #define **WLAN_TIMEOUT_REASSOC_DEADLINE** 1
- #define **WLAN_TIMEOUT_KEY_LIFETIME** 2
- #define **WLAN_TIMEOUT_ASSOC_COMEBACK** 3
- #define **IEEE80211_DA_FROMDS** addr1
- #define **IEEE80211_BSSID_FROMDS** addr2
- #define **IEEE80211_SA_FROMDS** addr3
- #define **IEEE80211_HDRLEN** (sizeof(struct ieee80211_hdr))

- #define **IEEE80211_FC**(type, stype) host_to_le16((type << 2) | (stype << 4))
- #define **ERP_INFO_NON_ERP_PRESENT** BIT(0)
- #define **ERP_INFO_USE_PROTECTION** BIT(1)
- #define **ERP_INFO_BARKER_PREAMBLE_MODE** BIT(2)
- #define **SET_2BIT_U8**(_ptr_, _shift_, _val_)
- #define **GET_2BIT_U8**(_var_, _shift_) (((_var_) & (((u8)3) << (_shift_))) >> (_shift_))
- #define **SET_2BIT_LE16**(_u16ptr_, _shift_, _val_)
- #define **GET_2BIT_LE16**(_var_, _shift_) (((_var_) & (((u16)3) << (_shift_))) >> (_shift_))
- #define **SET_2BIT_LE32**(_u32ptr_, _shift_, _val_)
- #define **GET_2BIT_LE32**(_var_, _shift_) (((_var_) & (((u32)3) << (_shift_))) >> (_shift_))
- #define **SET_3BIT_LE16**(_u16ptr_, _shift_, _val_)
- #define **GET_3BIT_LE16**(_var_, _shift_) (((_var_) & (((u16)7) << (_shift_))) >> (_shift_))
- #define **SET_3BIT_LE32**(_u32ptr_, _shift_, _val_)
- #define **GET_3BIT_LE32**(_var_, _shift_) (((_var_) & (((u32)7) << (_shift_))) >> (_shift_))
- #define **HT_CAP_INFO_LDPC_CODING_CAP** ((u16) BIT(0))
- #define **HT_CAP_INFO_SUPP_CHANNEL_WIDTH_SET** ((u16) BIT(1))
- #define **HT_CAP_INFO_SMPS_MASK** ((u16) (BIT(2) | BIT(3)))
- #define **HT_CAP_INFO_SMPS_STATIC** ((u16) 0)
- #define **HT_CAP_INFO_SMPS_DYNAMIC** ((u16) BIT(2))
- #define **HT_CAP_INFO_SMPS_DISABLED** ((u16) (BIT(2) | BIT(3)))
- #define **HT_CAP_INFO_GREEN_FIELD** ((u16) BIT(4))
- #define **HT_CAP_INFO_SHORT_GI20MHZ** ((u16) BIT(5))
- #define **HT_CAP_INFO_SHORT_GI40MHZ** ((u16) BIT(6))
- #define **HT_CAP_INFO_TX_STBC** ((u16) BIT(7))
- #define **HT_CAP_INFO_RX_STBC_MASK** ((u16) (BIT(8) | BIT(9)))
- #define **HT_CAP_INFO_RX_STBC_1** ((u16) BIT(8))
- #define **HT_CAP_INFO_RX_STBC_12** ((u16) BIT(9))
- #define **HT_CAP_INFO_RX_STBC_123** ((u16) (BIT(8) | BIT(9)))
- #define **HT_CAP_INFO_DELAYED_BA** ((u16) BIT(10))
- #define **HT_CAP_INFO_MAX_AMSDU_SIZE** ((u16) BIT(11))
- #define **HT_CAP_INFO_DSSS_CCK40MHZ** ((u16) BIT(12))
- #define **HT_CAP_INFO_PSMP_SUPP** ((u16) BIT(13))
- #define **HT_CAP_INFO_40MHZ_INTOLERANT** ((u16) BIT(14))
- #define **HT_CAP_INFO_LSIG_TXOP_PROTECT_SUPPORT** ((u16) BIT(15))
- #define **MAC_HT_PARAM_INFO_MAX_RX_AMPDU_FACTOR_OFFSET** 0
- #define **MAC_HT_PARAM_INFO_MAX_MPDU_DENSITY_OFFSET** 2
- #define **EXT_HT_CAP_INFO_PCO** ((u16) BIT(0))
- #define **EXT_HT_CAP_INFO_TRANS_TIME_OFFSET** 1
- #define **EXT_HT_CAP_INFO_MCS_FEEDBACK_OFFSET** 8
- #define **EXT_HT_CAP_INFO_HTC_SUPPORTED** ((u16) BIT(10))
- #define **EXT_HT_CAP_INFO_RD_RESPONDER** ((u16) BIT(11))
- #define **TX_BEAMFORM_CAP_TXBF_CAP** ((u32) BIT(0))
- #define **TX_BEAMFORM_CAP_RX_STAGGERED_SOUNDING_CAP** ((u32) BIT(1))
- #define **TX_BEAMFORM_CAP_TX_STAGGERED_SOUNDING_CAP** ((u32) BIT(2))
- #define **TX_BEAMFORM_CAP_RX_ZLF_CAP** ((u32) BIT(3))
- #define **TX_BEAMFORM_CAP_TX_ZLF_CAP** ((u32) BIT(4))
- #define **TX_BEAMFORM_CAP_IMPLICIT_ZLF_CAP** ((u32) BIT(5))
- #define **TX_BEAMFORM_CAP_CALIB_OFFSET** 6
- #define **TX_BEAMFORM_CAP_EXPLICIT_CSI_TXBF_CAP** ((u32) BIT(8))

- #define **TX_BEAMFORM_CAP_EXPLICIT_UNCOMPR_STEERING_MATRIX_CAP** ((u32) BIT(9))
- #define **TX_BEAMFORM_CAP_EXPLICIT_BF_CSI_FEEDBACK_CAP** ((u32) BIT(10))
- #define **TX_BEAMFORM_CAP_EXPLICIT_BF_CSI_FEEDBACK_OFFSET** 11
- #define **TX_BEAMFORM_CAP_EXPLICIT_UNCOMPR_STEERING_MATRIX_FEEDBACK_OFFSET** 13
- #define **TX_BEAMFORM_CAP_EXPLICIT_COMPRESSED_STEERING_MATRIX_FEEDBACK_OFFSET** 15
- #define **TX_BEAMFORM_CAP_MINIMAL_GROUPING_OFFSET** 17
- #define **TX_BEAMFORM_CAP_CSI_NUM_BEAMFORMER_ANT_OFFSET** 19
- #define **TX_BEAMFORM_CAP_UNCOMPRESSED_STEERING_MATRIX_BEAMFORMER_ANT_OFFSET** 21
- #define **TX_BEAMFORM_CAP_COMPRESSED_STEERING_MATRIX_BEAMFORMER_ANT_OFFSET** 23
- #define **TX_BEAMFORM_CAP_SCI_MAX_OF_ROWS_BEANFORMER_SUPPORTED_OFFSET** 25
- #define **ASEL_CAPABILITY_ASEL_CAPABLE** ((u8) BIT(0))
- #define **ASEL_CAPABILITY_EXPLICIT_CSI_FEEDBACK_BASED_TX_AS_CAP** ((u8) BIT(1))
- #define **ASEL_CAPABILITY_ANT_INDICES_FEEDBACK_BASED_TX_AS_CAP** ((u8) BIT(2))
- #define **ASEL_CAPABILITY_EXPLICIT_CSI_FEEDBACK_CAP** ((u8) BIT(3))
- #define **ASEL_CAPABILITY_ANT_INDICES_FEEDBACK_CAP** ((u8) BIT(4))
- #define **ASEL_CAPABILITY_RX_AS_CAP** ((u8) BIT(5))
- #define **ASEL_CAPABILITY_TX_SOUND_PPDUS_CAP** ((u8) BIT(6))
- #define **REC_TRANS_CHNL_WIDTH_20** 0
- #define **REC_TRANS_CHNL_WIDTH_ANY** 1
- #define **OP_MODE_PURE** 0
- #define **OP_MODE_MAY_BE_LEGACY_STAS** 1
- #define **OP_MODE_20MHZ_HT_STA_ASSOCED** 2
- #define **OP_MODE_MIXED** 3
- #define **HT_INFO_HT_PARAM_SECONDARY_CHNL_OFF_MASK** ((u8) BIT(0) | BIT(1))
- #define **HT_INFO_HT_PARAM_SECONDARY_CHNL_ABOVE** ((u8) BIT(0))
- #define **HT_INFO_HT_PARAM_SECONDARY_CHNL_BELOW** ((u8) BIT(0) | BIT(1))
- #define **HT_INFO_HT_PARAM_REC_TRANS_CHNL_WIDTH** ((u8) BIT(2))
- #define **HT_INFO_HT_PARAM_RIFS_MODE** ((u8) BIT(3))
- #define **HT_INFO_HT_PARAM_CTRL_ACCESS_ONLY** ((u8) BIT(4))
- #define **HT_INFO_HT_PARAM_SRV_INTERVAL_GRANULARITY** ((u8) BIT(5))
- #define **HT_INFO_OPERATION_MODE_OP_MODE_MASK** ((le16) (0x0001 | 0x0002))
- #define **HT_INFO_OPERATION_MODE_OP_MODE_OFFSET** 0
- #define **HT_INFO_OPERATION_MODE_NON_GF_DEVS_PRESENT** ((u8) BIT(2))
- #define **HT_INFO_OPERATION_MODE_TRANSMIT_BURST_LIMIT** ((u8) BIT(3))
- #define **HT_INFO_OPERATION_MODE_NON_HT_STA_PRESENT** ((u8) BIT(4))
- #define **HT_INFO_STBC_PARAM_DUAL_BEACON** ((u16) BIT(6))
- #define **HT_INFO_STBC_PARAM_DUAL_STBC_PROTECT** ((u16) BIT(7))
- #define **HT_INFO_STBC_PARAM_SECONDARY_BCN** ((u16) BIT(8))
- #define **HT_INFO_STBC_PARAM_LSIG_TXOP_PROTECT_ALLOWED** ((u16) BIT(9))
- #define **HT_INFO_STBC_PARAM_PCO_ACTIVE** ((u16) BIT(10))
- #define **HT_INFO_STBC_PARAM_PCO_PHASE** ((u16) BIT(11))
- #define **SECONDARY_CHANNEL_OFFSET_NONE** 0
- #define **SECONDARY_CHANNEL_OFFSET_ABOVE** 1

- #define **SECONDARY_CHANNEL_OFFSET_BELOW** 3
- #define **CHANNEL_WIDTH_20** 0
- #define **CHANNEL_WIDTH_ANY** 1
- #define **PWR_SAVE_MODE_STATIC** 0
- #define **PWR_SAVE_MODE_DYNAMIC** 1
- #define **OUI_MICROSOFT** 0x0050f2
- #define **WMM_OUI_TYPE** 2
- #define **WMM_OUI_SUBTYPE_INFORMATION_ELEMENT** 0
- #define **WMM_OUI_SUBTYPE_PARAMETER_ELEMENT** 1
- #define **WMM_OUI_SUBTYPE_TSPEC_ELEMENT** 2
- #define **WMM_VERSION** 1
- #define **WMM_ACTION_CODE_ADDTS_REQ** 0
- #define **WMM_ACTION_CODE_ADDTS_RESP** 1
- #define **WMM_ACTION_CODE_DELTS** 2
- #define **WMM_ADDTS_STATUS_ADMISSION_ACCEPTED** 0
- #define **WMM_ADDTS_STATUS_INVALID_PARAMETERS** 1
- #define **WMM_ADDTS_STATUS_REFUSED** 3
- #define **WMM_TSPEC_DIRECTION_UPLINK** 0
- #define **WMM_TSPEC_DIRECTION_DOWNLINK** 1
- #define **WMM_TSPEC_DIRECTION_BI_DIRECTIONAL** 3
- #define **WMM_AC_AIFSN_MASK** 0x0f
- #define **WMM_AC_AIFNS_SHIFT** 0
- #define **WMM_AC_ACM** 0x10
- #define **WMM_AC_ACI_MASK** 0x60
- #define **WMM_AC_ACI_SHIFT** 5
- #define **WMM_AC_ECWMIN_MASK** 0x0f
- #define **WMM_AC_ECWMIN_SHIFT** 0
- #define **WMM_AC_ECWMAX_MASK** 0xf0
- #define **WMM_AC_ECWMAX_SHIFT** 4
- #define **OUI_BROADCOM** 0x00904c
- #define **VENDOR_HT_CAPAB_OUI_TYPE** 0x33
- #define **WLAN_CIPHER_SUITE_USE_GROUP** 0x000FAC00
- #define **WLAN_CIPHER_SUITE_WEP40** 0x000FAC01
- #define **WLAN_CIPHER_SUITE_TKIP** 0x000FAC02
- #define **WLAN_CIPHER_SUITE_CCMP** 0x000FAC04
- #define **WLAN_CIPHER_SUITE_WEP104** 0x000FAC05
- #define **WLAN_CIPHER_SUITE_AES_CMAC** 0x000FAC06
- #define **WLAN_AKM_SUITE_8021X** 0x000FAC01
- #define **WLAN_AKM_SUITE_PSK** 0x000FAC02

Enumerations

- enum { **MAX_RX_AMPDU_FACTOR_8KB** = 0, **MAX_RX_AMPDU_FACTOR_16KB**, **MAX_RX_AMPDU_FACTOR_32KB**, **MAX_RX_AMPDU_FACTOR_64KB** }
- enum { **CALIBRATION_NOT_SUPPORTED** = 0, **CALIBRATION_CANNOT_INIT**, **CALIBRATION_CAN_INIT**, **CALIBRATION_FULL_SUPPORT** }
- enum { **MCS_FEEDBACK_NOT_PROVIDED** = 0, **MCS_FEEDBACK_UNSOLICITED**, **MCS_FEEDBACK_MRQ_RESPONSE** }
- enum { **WMM_AC_BE** = 0, **WMM_AC_BK** = 1, **WMM_AC_VI** = 2, **WMM_AC_VO** = 3 }

Variables

- struct ieee80211_hdr **STRUCT_PACKED**

15.76.1 Detailed Description

IEEE 802.11 Frame type definitions.

Copyright

Copyright (c) 2002-2007, Jouni Malinen <j@w1.fi> Copyright (c) 2007-2008 Intel Corporation

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

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

See README and COPYING for more details.

15.76.2 Define Documentation

15.76.2.1 #define SET_2BIT_LE16(_u16ptr_, _shift_, _val_)

Value:

```
((*(_u16ptr_) &= ~(3 << (_shift_))),                              \
         (*(_u16ptr_) |=                                                \
                (((*(_u16ptr_)) & (((u16)3) << ((u16)_shift_))) |       \
                (((u16)(_val_) & (u16)3) << (u16)(_shift_)))))
```

15.76.2.2 #define SET_2BIT_LE32(_u32ptr_, _shift_, _val_)

Value:

```
((*(_u32ptr_) &= ~(3 << (_shift_))),                              \
         (*(_u32ptr_) |= (((*(_u32ptr_)) & (((u32)3) << (_shift_))) |   \
                         (((u32)(_val_) & 3) << _shift_))))
```

15.76.2.3 #define SET_2BIT_U8(_ptr_, _shift_, _val_)

Value:

```
((*(_ptr_) &= ~(3 << (_shift_))),                                 \
         (*(_ptr_) |= (*(_ptr_) & (((u8)3) << (_shift_))) |             \
                      (((u8)(_val_) & 3) << _shift_)))
```

15.76.2.4 #define SET_3BIT_LE16(_u16ptr_, _shift_, _val_)

Value:

```
((*(_u16ptr_) &= ~(7 << (_shift_))),                              \
        (*(_u16ptr_) |= (((*(_u16ptr_)) & (((u16)7) << (_shift_))) |    \
                        (((u16)(_val_) & 7) << _shift_))))
```

15.76.2.5 #define SET_3BIT_LE32(_u32ptr_, _shift_, _val_)

Value:

```
((*(_u32ptr_) &= ~(7 << (_shift_))),                                \
         (*(_u32ptr_) |= (((*(_u32ptr_)) & (((u32)7) << (_shift_))) |   \
                         (((u32)(_val_) & 7) << _shift_))))
```

15.77 src/common/privsep_commands.h File Reference

WPA Supplicant - privilege separation commands.

Data Structures

- struct privsep_cmd_associate
- struct privsep_cmd_set_key

Enumerations

- enum **privsep_cmd** {

  PRIVSEP_CMD_REGISTER, **PRIVSEP_CMD_UNREGISTER**, **PRIVSEP_CMD_SCAN**, **PRIVSEP_CMD_GET_SCAN_RESULTS**,

  PRIVSEP_CMD_ASSOCIATE, **PRIVSEP_CMD_GET_BSSID**, **PRIVSEP_CMD_GET_SSID**, **PRIVSEP_CMD_SET_KEY**,

  PRIVSEP_CMD_GET_CAPA, **PRIVSEP_CMD_L2_REGISTER**, **PRIVSEP_CMD_L2_UNREGISTER**, **PRIVSEP_CMD_L2_NOTIFY_AUTH_START**,

  PRIVSEP_CMD_L2_SEND, **PRIVSEP_CMD_SET_COUNTRY** }
- enum **privsep_event** {

  PRIVSEP_EVENT_SCAN_RESULTS, **PRIVSEP_EVENT_ASSOC**, **PRIVSEP_EVENT_DISASSOC**, **PRIVSEP_EVENT_ASSOCINFO**,

  PRIVSEP_EVENT_MICHAEL_MIC_FAILURE, **PRIVSEP_EVENT_INTERFACE_STATUS**, **PRIVSEP_EVENT_PMKID_CANDIDATE**, **PRIVSEP_EVENT_STKSTART**,

  PRIVSEP_EVENT_FT_RESPONSE, **PRIVSEP_EVENT_RX_EAPOL**, **PRIVSEP_EVENT_STA_RX** }

15.77.1 Detailed Description

WPA Supplicant - privilege separation commands.

Copyright

Copyright (c) 2007-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.78 src/common/wpa_common.c File Reference

WPA/RSN - Shared functions for supplicant and authenticator. `#include "includes.h"`

```
#include "common.h"
#include "md5.h"
#include "sha1.h"
#include "sha256.h"
#include "aes_wrap.h"
#include "crypto.h"
#include "ieee802_11_defs.h"
#include "defs.h"
#include "wpa_common.h"
```

Functions

- int wpa_eapol_key_mic (const u8 *key, int ver, const u8 *buf, size_t len, u8 *mic)

  Calculate EAPOL-Key MIC.

- void wpa_pmk_to_ptk (const u8 *pmk, size_t pmk_len, const char *label, const u8 *addr1, const u8 *addr2, const u8 *nonce1, const u8 *nonce2, u8 *ptk, size_t ptk_len, int use_sha256)

  Calculate PTK from PMK, addresses, and nonces.

- int wpa_parse_wpa_ie_rsn (const u8 *rsn_ie, size_t rsn_ie_len, struct wpa_ie_data *data)

  Parse RSN IE.

- void rsn_pmkid (const u8 *pmk, size_t pmk_len, const u8 *aa, const u8 *spa, u8 *pmkid, int use_sha256)

  Calculate PMK identifier.

- const char * wpa_cipher_txt (int cipher)

  Convert cipher suite to a text string.

- const char * wpa_key_mgmt_txt (int key_mgmt, int proto)

  Convert key management suite to a text string.

15.78.1 Detailed Description

WPA/RSN - Shared functions for supplicant and authenticator.

Copyright

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

See README and COPYING for more details.

15.78.2 Function Documentation

15.78.2.1 void rsn_pmkid (const u8 * *pmk*, size_t *pmk_len*, const u8 * *aa*, const u8 * *spa*, u8 * *pmkid*, int *use_sha256*)

Calculate PMK identifier.

Parameters:

pmk Pairwise master key

pmk_len Length of pmk in bytes

aa Authenticator address

spa Supplicant address

pmkid Buffer for PMKID

use_sha256 Whether to use SHA256-based KDF

IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy PMKID = HMAC-SHA1-128(PMK, "PMK Name" || AA || SPA)

15.78.2.2 const char* wpa_cipher_txt (int *cipher*)

Convert cipher suite to a text string.

Parameters:

cipher Cipher suite (WPA_CIPHER_* enum)

Returns:

Pointer to a text string of the cipher suite name

15.78.2.3 int wpa_eapol_key_mic (const u8 * *key*, int *ver*, const u8 * *buf*, size_t *len*, u8 * *mic*)

Calculate EAPOL-Key MIC.

Parameters:

key EAPOL-Key Key Confirmation Key (KCK)

ver Key descriptor version (WPA_KEY_INFO_TYPE_*)

buf Pointer to the beginning of the EAPOL header (version field)

len Length of the EAPOL frame (from EAPOL header to the end of the frame)

mic Pointer to the buffer to which the EAPOL-Key MIC is written

Returns:

0 on success, -1 on failure

Calculate EAPOL-Key MIC for an EAPOL-Key packet. The EAPOL-Key MIC field has to be cleared (all zeroes) when calling this function.

Note: 'IEEE Std 802.11i-2004 - 8.5.2 EAPOL-Key frames' has an error in the description of the Key MIC calculation. It includes packet data from the beginning of the EAPOL-Key header, not EAPOL header. This incorrect change happened during final editing of the standard and the correct behavior is defined in the last draft (IEEE 802.11i/D10).

15.78.2.4 const char* wpa_key_mgmt_txt (int *key_mgmt*, int *proto*)

Convert key management suite to a text string.

Parameters:

key_mgmt Key management suite (WPA_KEY_MGMT_* enum)

proto WPA/WPA2 version (WPA_PROTO_*)

Returns:

Pointer to a text string of the key management suite name

15.78.2.5 int wpa_parse_wpa_ie_rsn (const u8 * *rsn_ie*, size_t *rsn_ie_len*, struct wpa_ie_data * *data*)

Parse RSN IE.

Parameters:

rsn_ie Buffer containing RSN IE

rsn_ie_len RSN IE buffer length (including IE number and length octets)

data Pointer to structure that will be filled in with parsed data

Returns:

0 on success, <0 on failure

15.78.2.6 void wpa_pmk_to_ptk (const u8 * *pmk*, size_t *pmk_len*, const char * *label*, const u8 * *addr1*, const u8 * *addr2*, const u8 * *nonce1*, const u8 * *nonce2*, u8 * *ptk*, size_t *ptk_len*, int *use_sha256*)

Calculate PTK from PMK, addresses, and nonces.

Parameters:

pmk Pairwise master key

pmk_len Length of PMK

label Label to use in derivation

addr1 AA or SA

addr2 SA or AA

nonce1 ANonce or SNonce

nonce2 SNonce or ANonce

ptk Buffer for pairwise transient key

ptk_len Length of PTK

use_sha256 Whether to use SHA256-based KDF

IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy PTK = PRF-X(PMK, "Pairwise key expansion", Min(AA, SA) || Max(AA, SA) || Min(ANonce, SNonce) || Max(ANonce, SNonce))

STK = PRF-X(SMK, "Peer key expansion", Min(MAC_I, MAC_P) || Max(MAC_I, MAC_P) || Min(INonce, PNonce) || Max(INonce, PNonce))

15.79 src/common/wpa_common.h File Reference

WPA definitions shared between hostapd and wpa_supplicant.

Data Structures

- struct wpa_eapol_key
- struct wpa_ptk

  WPA Pairwise Transient Key.

- struct wpa_ie_hdr
- struct rsn_ie_hdr
- struct rsn_error_kde
- struct wpa_ie_data

Defines

- #define **WPA_MAX_SSID_LEN** 32
- #define **PMKID_LEN** 16
- #define **PMK_LEN** 32
- #define **WPA_REPLAY_COUNTER_LEN** 8
- #define **WPA_NONCE_LEN** 32
- #define **WPA_KEY_RSC_LEN** 8
- #define **WPA_GMK_LEN** 32
- #define **WPA_GTK_MAX_LEN** 32
- #define **WPA_SELECTOR_LEN** 4
- #define **WPA_VERSION** 1
- #define **RSN_SELECTOR_LEN** 4
- #define **RSN_VERSION** 1
- #define **RSN_SELECTOR**(a, b, c, d)
- #define **WPA_AUTH_KEY_MGMT_NONE** RSN_SELECTOR(0x00, 0x50, 0xf2, 0)
- #define **WPA_AUTH_KEY_MGMT_UNSPEC_802_1X** RSN_SELECTOR(0x00, 0x50, 0xf2, 1)
- #define **WPA_AUTH_KEY_MGMT_PSK_OVER_802_1X** RSN_SELECTOR(0x00, 0x50, 0xf2, 2)
- #define **WPA_CIPHER_SUITE_NONE** RSN_SELECTOR(0x00, 0x50, 0xf2, 0)
- #define **WPA_CIPHER_SUITE_WEP40** RSN_SELECTOR(0x00, 0x50, 0xf2, 1)
- #define **WPA_CIPHER_SUITE_TKIP** RSN_SELECTOR(0x00, 0x50, 0xf2, 2)
- #define **WPA_CIPHER_SUITE_CCMP** RSN_SELECTOR(0x00, 0x50, 0xf2, 4)
- #define **WPA_CIPHER_SUITE_WEP104** RSN_SELECTOR(0x00, 0x50, 0xf2, 5)
- #define **RSN_AUTH_KEY_MGMT_UNSPEC_802_1X** RSN_SELECTOR(0x00, 0x0f, 0xac, 1)
- #define **RSN_AUTH_KEY_MGMT_PSK_OVER_802_1X** RSN_SELECTOR(0x00, 0x0f, 0xac, 2)
- #define **RSN_AUTH_KEY_MGMT_802_1X_SHA256** RSN_SELECTOR(0x00, 0x0f, 0xac, 5)
- #define **RSN_AUTH_KEY_MGMT_PSK_SHA256** RSN_SELECTOR(0x00, 0x0f, 0xac, 6)
- #define **RSN_CIPHER_SUITE_NONE** RSN_SELECTOR(0x00, 0x0f, 0xac, 0)
- #define **RSN_CIPHER_SUITE_WEP40** RSN_SELECTOR(0x00, 0x0f, 0xac, 1)
- #define **RSN_CIPHER_SUITE_TKIP** RSN_SELECTOR(0x00, 0x0f, 0xac, 2)
- #define **RSN_CIPHER_SUITE_CCMP** RSN_SELECTOR(0x00, 0x0f, 0xac, 4)
- #define **RSN_CIPHER_SUITE_WEP104** RSN_SELECTOR(0x00, 0x0f, 0xac, 5)

- #define **RSN_KEY_DATA_GROUPKEY** RSN_SELECTOR(0x00, 0x0f, 0xac, 1)
- #define **RSN_KEY_DATA_MAC_ADDR** RSN_SELECTOR(0x00, 0x0f, 0xac, 3)
- #define **RSN_KEY_DATA_PMKID** RSN_SELECTOR(0x00, 0x0f, 0xac, 4)
- #define **WPA_OUI_TYPE** RSN_SELECTOR(0x00, 0x50, 0xf2, 1)
- #define **RSN_SELECTOR_PUT**(a, val) WPA_PUT_BE32((u8 *) (a), (val))
- #define **RSN_SELECTOR_GET**(a) WPA_GET_BE32((const u8 *) (a))
- #define **RSN_NUM_REPLAY_COUNTERS_1** 0
- #define **RSN_NUM_REPLAY_COUNTERS_2** 1
- #define **RSN_NUM_REPLAY_COUNTERS_4** 2
- #define **RSN_NUM_REPLAY_COUNTERS_16** 3
- #define **WPA_CAPABILITY_PREAUTH** BIT(0)
- #define **WPA_CAPABILITY_NO_PAIRWISE** BIT(1)
- #define **WPA_CAPABILITY_MFPR** BIT(6)
- #define **WPA_CAPABILITY_MFPC** BIT(7)
- #define **WPA_CAPABILITY_PEERKEY_ENABLED** BIT(9)
- #define **MOBILITY_DOMAIN_ID_LEN** 2
- #define **FT_R0KH_ID_MAX_LEN** 48
- #define **FT_R1KH_ID_LEN** 6
- #define **WPA_PMK_NAME_LEN** 16
- #define **WPA_KEY_INFO_TYPE_MASK** ((u16) (BIT(0) | BIT(1) | BIT(2)))
- #define **WPA_KEY_INFO_TYPE_HMAC_MD5_RC4** BIT(0)
- #define **WPA_KEY_INFO_TYPE_HMAC_SHA1_AES** BIT(1)
- #define **WPA_KEY_INFO_TYPE_AES_128_CMAC** 3
- #define **WPA_KEY_INFO_KEY_TYPE** BIT(3)
- #define **WPA_KEY_INFO_KEY_INDEX_MASK** (BIT(4) | BIT(5))
- #define **WPA_KEY_INFO_KEY_INDEX_SHIFT** 4
- #define **WPA_KEY_INFO_INSTALL** BIT(6)
- #define **WPA_KEY_INFO_TXRX** BIT(6)
- #define **WPA_KEY_INFO_ACK** BIT(7)
- #define **WPA_KEY_INFO_MIC** BIT(8)
- #define **WPA_KEY_INFO_SECURE** BIT(9)
- #define **WPA_KEY_INFO_ERROR** BIT(10)
- #define **WPA_KEY_INFO_REQUEST** BIT(11)
- #define **WPA_KEY_INFO_ENCR_KEY_DATA** BIT(12)
- #define **WPA_KEY_INFO_SMK_MESSAGE** BIT(13)

Functions

- int wpa_eapol_key_mic (const u8 *key, int ver, const u8 *buf, size_t len, u8 *mic)

  Calculate EAPOL-Key MIC.

- void wpa_pmk_to_ptk (const u8 *pmk, size_t pmk_len, const char *label, const u8 *addr1, const u8 *addr2, const u8 *nonce1, const u8 *nonce2, u8 *ptk, size_t ptk_len, int use_sha256)

  Calculate PTK from PMK, addresses, and nonces.

- int wpa_parse_wpa_ie_rsn (const u8 *rsn_ie, size_t rsn_ie_len, struct wpa_ie_data *data)

  Parse RSN IE.

- void rsn_pmkid (const u8 *pmk, size_t pmk_len, const u8 *aa, const u8 *spa, u8 *pmkid, int use_sha256)

Calculate PMK identifier.

- const char * wpa_cipher_txt (int cipher)

  Convert cipher suite to a text string.

- const char * wpa_key_mgmt_txt (int key_mgmt, int proto)

  Convert key management suite to a text string.

Variables

- struct wpa_eapol_key STRUCT_PACKED

  WPA Pairwise Transient Key.

15.79.1 Detailed Description

WPA definitions shared between hostapd and wpa_supplicant.

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.79.2 Define Documentation

15.79.2.1 #define RSN_SELECTOR(a, b, c, d)

Value:

```
((((u32) (a)) << 24) | (((u32) (b)) << 16) | (((u32) (c)) << 8) | \
         (u32) (d))
```

15.79.3 Function Documentation

15.79.3.1 void rsn_pmkid (const u8 * *pmk*, size_t *pmk_len*, const u8 * *aa*, const u8 * *spa*, u8 * *pmkid*, int *use_sha256*)

Calculate PMK identifier.

Parameters:

pmk Pairwise master key

pmk_len Length of pmk in bytes

aa Authenticator address

spa Supplicant address
pmkid Buffer for PMKID
use_sha256 Whether to use SHA256-based KDF

IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy PMKID = HMAC-SHA1-128(PMK, "PMK Name" || AA || SPA)

15.79.3.2 const char* wpa_cipher_txt (int *cipher*)

Convert cipher suite to a text string.

Parameters:

cipher Cipher suite (WPA_CIPHER_* enum)

Returns:

Pointer to a text string of the cipher suite name

15.79.3.3 int wpa_eapol_key_mic (const u8 * *key*, int *ver*, const u8 * *buf*, size_t *len*, u8 * *mic*)

Calculate EAPOL-Key MIC.

Parameters:

key EAPOL-Key Key Confirmation Key (KCK)
ver Key descriptor version (WPA_KEY_INFO_TYPE_*)
buf Pointer to the beginning of the EAPOL header (version field)
len Length of the EAPOL frame (from EAPOL header to the end of the frame)
mic Pointer to the buffer to which the EAPOL-Key MIC is written

Returns:

0 on success, -1 on failure

Calculate EAPOL-Key MIC for an EAPOL-Key packet. The EAPOL-Key MIC field has to be cleared (all zeroes) when calling this function.

Note: 'IEEE Std 802.11i-2004 - 8.5.2 EAPOL-Key frames' has an error in the description of the Key MIC calculation. It includes packet data from the beginning of the EAPOL-Key header, not EAPOL header. This incorrect change happened during final editing of the standard and the correct behavior is defined in the last draft (IEEE 802.11i/D10).

15.79.3.4 const char* wpa_key_mgmt_txt (int *key_mgmt*, int *proto*)

Convert key management suite to a text string.

Parameters:

key_mgmt Key management suite (WPA_KEY_MGMT_* enum)
proto WPA/WPA2 version (WPA_PROTO_*)

Returns:

Pointer to a text string of the key management suite name

15.79.3.5 int wpa_parse_wpa_ie_rsn (const u8 * *rsn_ie*, size_t *rsn_ie_len*, struct wpa_ie_data * *data*)

Parse RSN IE.

Parameters:

rsn_ie Buffer containing RSN IE

rsn_ie_len RSN IE buffer length (including IE number and length octets)

data Pointer to structure that will be filled in with parsed data

Returns:

0 on success, <0 on failure

15.79.3.6 void wpa_pmk_to_ptk (const u8 * *pmk*, size_t *pmk_len*, const char * *label*, const u8 * *addr1*, const u8 * *addr2*, const u8 * *nonce1*, const u8 * *nonce2*, u8 * *ptk*, size_t *ptk_len*, int *use_sha256*)

Calculate PTK from PMK, addresses, and nonces.

Parameters:

pmk Pairwise master key

pmk_len Length of PMK

label Label to use in derivation

addr1 AA or SA

addr2 SA or AA

nonce1 ANonce or SNonce

nonce2 SNonce or ANonce

ptk Buffer for pairwise transient key

ptk_len Length of PTK

use_sha256 Whether to use SHA256-based KDF

IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy PTK = PRF-X(PMK, "Pairwise key expansion", Min(AA, SA) || Max(AA, SA) || Min(ANonce, SNonce) || Max(ANonce, SNonce))

STK = PRF-X(SMK, "Peer key expansion", Min(MAC_I, MAC_P) || Max(MAC_I, MAC_P) || Min(INonce, PNonce) || Max(INonce, PNonce))

15.79.4 Variable Documentation

15.79.4.1 struct rsn_error_kde STRUCT_PACKED

WPA Pairwise Transient Key. IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy

15.80 src/common/wpa_ctrl.c File Reference

wpa_supplicant/hostapd control interface library `#include "includes.h"`

`#include "wpa_ctrl.h"`

`#include "common.h"`

Data Structures

- struct wpa_ctrl
  Internal structure for control interface library.

Functions

- int wpa_ctrl_attach (struct wpa_ctrl *ctrl)
  Register as an event monitor for the control interface.
- int wpa_ctrl_detach (struct wpa_ctrl *ctrl)
  Unregister event monitor from the control interface.

15.80.1 Detailed Description

wpa_supplicant/hostapd control interface library

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.80.2 Function Documentation

15.80.2.1 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().

15.80.2.2 int wpa_ctrl_detach (struct wpa_ctrl * *ctrl*)

Unregister event monitor from the control interface.

Parameters:

ctrl Control interface data from wpa_ctrl_open()

Returns:

0 on success, -1 on failure, -2 on timeout

This function unregisters the control interface connection as a monitor for wpa_supplicant/hostapd events, i.e., cancels the registration done with wpa_ctrl_attach().

15.81 src/common/wpa_ctrl.h File Reference

wpa_supplicant/hostapd control interface library

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 "
- #define WPA_EVENT_SCAN_RESULTS "CTRL-EVENT-SCAN-RESULTS "
- #define WPS_EVENT_OVERLAP "WPS-OVERLAP-DETECTED "
- #define WPS_EVENT_AP_AVAILABLE_PBC "WPS-AP-AVAILABLE-PBC "
- #define WPS_EVENT_AP_AVAILABLE_PIN "WPS-AP-AVAILABLE-PIN "
- #define WPS_EVENT_AP_AVAILABLE "WPS-AP-AVAILABLE "
- #define WPS_EVENT_CRED_RECEIVED "WPS-CRED-RECEIVED "
- #define WPS_EVENT_M2D "WPS-M2D "
- #define WPS_EVENT_FAIL "WPS-FAIL "
- #define WPS_EVENT_SUCCESS "WPS-SUCCESS "
- #define WPS_EVENT_TIMEOUT "WPS-TIMEOUT "
- #define **WPS_EVENT_ER_AP_ADD** "WPS-ER-AP-ADD "
- #define **WPS_EVENT_ER_AP_REMOVE** "WPS-ER-AP-REMOVE "
- #define **WPS_EVENT_ER_ENROLLEE_ADD** "WPS-ER-ENROLLEE-ADD "
- #define **WPS_EVENT_ER_ENROLLEE_REMOVE** "WPS-ER-ENROLLEE-REMOVE "
- #define **WPS_EVENT_PIN_NEEDED** "WPS-PIN-NEEDED "
- #define **WPS_EVENT_NEW_AP_SETTINGS** "WPS-NEW-AP-SETTINGS "
- #define **WPS_EVENT_REG_SUCCESS** "WPS-REG-SUCCESS "
- #define **WPS_EVENT_AP_SETUP_LOCKED** "WPS-AP-SETUP-LOCKED "
- #define **AP_STA_CONNECTED** "AP-STA-CONNECTED "
- #define **AP_STA_DISCONNECTED** "AP-STA-DISCONNECTED "

Functions

- struct 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.

15.81.1 Detailed Description

wpa_supplicant/hostapd control interface library

Copyright

Copyright (c) 2004-2006, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.81.2 Define Documentation

15.81.2.1 #define WPA_CTRL_REQ "CTRL-REQ-"

Interactive request for identity/password/pin

15.81.2.2 #define WPA_CTRL_RSP "CTRL-RSP-"

Response to identity/password/pin request

15.81.2.3 #define WPA_EVENT_CONNECTED "CTRL-EVENT-CONNECTED "

Authentication completed successfully and data connection enabled

15.81.2.4 #define WPA_EVENT_DISCONNECTED "CTRL-EVENT-DISCONNECTED "

Disconnected, data connection is not available

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

EAP authentication failed (EAP-Failure received)

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

EAP method selected

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

EAP-Request/Notification received

15.81.2.8 #define WPA_EVENT_EAP_STARTED "CTRL-EVENT-EAP-STARTED "

EAP authentication started (EAP-Request/Identity received)

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

EAP authentication completed successfully

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

Password change was completed successfully

15.81.2.11 #define WPA_EVENT_SCAN_RESULTS "CTRL-EVENT-SCAN-RESULTS "

New scan results available

15.81.2.12 #define WPA_EVENT_TERMINATING "CTRL-EVENT-TERMINATING "

wpa_supplicant is exiting

15.81.2.13 #define WPS_EVENT_AP_AVAILABLE "WPS-AP-AVAILABLE "

Available WPS AP found in scan results

15.81.2.14 #define WPS_EVENT_AP_AVAILABLE_PBC "WPS-AP-AVAILABLE-PBC "

Available WPS AP with active PBC found in scan results

15.81.2.15 #define WPS_EVENT_AP_AVAILABLE_PIN "WPS-AP-AVAILABLE-PIN "

Available WPS AP with recently selected PIN registrar found in scan results

15.81.2.16 #define WPS_EVENT_CRED_RECEIVED "WPS-CRED-RECEIVED "

A new credential received

15.81.2.17 #define WPS_EVENT_FAIL "WPS-FAIL "

WPS registration failed after M2/M2D

15.81.2.18 #define WPS_EVENT_M2D "WPS-M2D "

M2D received

15.81.2.19 #define WPS_EVENT_OVERLAP "WPS-OVERLAP-DETECTED "

WPS overlap detected in PBC mode

15.81.2.20 #define WPS_EVENT_SUCCESS "WPS-SUCCESS "

WPS registration completed successfully

15.81.2.21 #define WPS_EVENT_TIMEOUT "WPS-TIMEOUT "

WPS enrollment attempt timed out and was terminated

15.81.3 Function Documentation

15.81.3.1 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().

15.81.3.2 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.

15.81.3.3 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().

15.81.3.4 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.

15.81.3.5 struct wpa_ctrl* wpa_ctrl_open (const char * *ctrl_path*) `[read]`

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.

15.81.3.6 int wpa_ctrl_pending (struct wpa_ctrl * *ctrl*)

Check whether there are pending event messages.

Parameters:

ctrl Control interface data from wpa_ctrl_open()

Returns:

1 if there are pending messages, 0 if no, or -1 on error

This function will check whether there are any pending control interface message available to be received with wpa_ctrl_recv(). wpa_ctrl_pending() is only used for event messages, i.e., wpa_ctrl_attach() must have been used to register the control interface as an event monitor.

15.81.3.7 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.

15.81.3.8 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.

15.82 src/crypto/aes-cbc.c File Reference

AES-128 CBC. `#include "includes.h"`

`#include "common.h"`

`#include "aes.h"`

`#include "aes_wrap.h"`

Functions

- 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.

15.82.1 Detailed Description

AES-128 CBC.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.82.2 Function Documentation

15.82.2.1 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

15.82.2.2 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

15.83 src/crypto/aes-ctr.c File Reference

AES-128 CTR. `#include "includes.h"`

`#include "common.h"`

`#include "aes.h"`

`#include "aes_wrap.h"`

Functions

- int aes_128_ctr_encrypt (const u8 *key, const u8 *nonce, u8 *data, size_t data_len)
  AES-128 CTR mode encryption.

15.83.1 Detailed Description

AES-128 CTR.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.83.2 Function Documentation

15.83.2.1 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

15.84 src/crypto/aes-eax.c File Reference

AES-128 EAX. `#include "includes.h"`

`#include "common.h"`

`#include "aes.h"`

`#include "aes_wrap.h"`

Functions

- 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.

15.84.1 Detailed Description

AES-128 EAX.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.84.2 Function Documentation

15.84.2.1 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

15.84.2.2 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

15.85 src/crypto/aes-encblock.c File Reference

AES encrypt_block. `#include "includes.h"`

`#include "common.h"`

`#include "aes.h"`

`#include "aes_wrap.h"`

Functions

- int aes_128_encrypt_block (const u8 *key, const u8 *in, u8 *out)
  Perform one AES 128-bit block operation.

15.85.1 Detailed Description

AES encrypt_block.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.85.2 Function Documentation

15.85.2.1 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

15.86 src/crypto/aes-internal-dec.c File Reference

AES (Rijndael) cipher - decrypt. `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

`#include "aes_i.h"`

Defines

- #define **ROUND**(i, d, s)

Functions

- void rijndaelKeySetupDec (u32 rk[ ], const u8 cipherKey[ ])
- 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.

15.86.1 Detailed Description

AES (Rijndael) cipher - decrypt. 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 <j@w1.fi>

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

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

See README and COPYING for more details.

15.86.2 Define Documentation

15.86.2.1 #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]
```

15.86.3 Function Documentation

15.86.3.1 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)

15.86.3.2 void aes_decrypt_deinit (void * *ctx*)

Deinitialize AES decryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.86.3.3 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

15.86.3.4 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.

15.87 src/crypto/aes-internal-enc.c File Reference

AES (Rijndael) cipher - encrypt. `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

`#include "aes_i.h"`

Defines

- #define **ROUND**(i, d, s)

Functions

- void **rijndaelEncrypt** (const u32 rk[ ], const u8 pt[16], u8 ct[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.

15.87.1 Detailed Description

AES (Rijndael) cipher - encrypt. 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 <j@w1.fi>

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

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

See README and COPYING for more details.

15.87.2 Define Documentation

15.87.2.1 #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]
```

15.87.3 Function Documentation

15.87.3.1 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)

15.87.3.2 void aes_encrypt_deinit (void * *ctx*)

Deinitialize AES encryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.87.3.3 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

15.88 src/crypto/aes-internal.c File Reference

AES (Rijndael) cipher. `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

`#include "aes_i.h"`

Functions

- void rijndaelKeySetupEnc (u32 rk[ ], const u8 cipherKey[ ])

Variables

- const u32 **Te0** [256]
- const u32 **Te1** [256]
- const u32 **Te2** [256]
- const u32 **Te3** [256]
- const u32 **Te4** [256]
- const u32 **Td0** [256]
- const u32 **Td1** [256]
- const u32 **Td2** [256]
- const u32 **Td3** [256]
- const u32 **Td4** [256]
- const u32 **rcon** [ ]

15.88.1 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

15.88.2 Function Documentation

15.88.2.1 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.

15.88.3 Variable Documentation

15.88.3.1 const u32 rcon[ ]

Initial value:

```
 {
        0x01000000, 0x02000000, 0x04000000, 0x08000000,
        0x10000000, 0x20000000, 0x40000000, 0x80000000,
        0x1B000000, 0x36000000,
}
```

15.89 src/crypto/aes-omac1.c File Reference

One-key CBC MAC (OMAC1) hash with AES-128. `#include "includes.h"`

`#include "common.h"`

`#include "aes.h"`

`#include "aes_wrap.h"`

Functions

- int omac1_aes_128_vector (const u8 *key, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

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

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

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

15.89.1 Detailed Description

One-key CBC MAC (OMAC1) hash with AES-128.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.89.2 Function Documentation

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

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

Parameters:

key 128-bit key for the hash operation

data Data buffer for which a MAC is determined

data_len Length of data buffer in bytes

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

Returns:

0 on success, -1 on failure

This is a mode for using block cipher (AES in this case) for authentication. OMAC1 was standardized with the name CMAC by NIST in a Special Publication (SP) 800-38B.

15.89.2.2 int omac1_aes_128_vector (const u8 * *key*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

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

Parameters:

key 128-bit key for the hash operation

num_elem Number of elements in the data vector

addr Pointers to the data areas

len Lengths of the data blocks

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

Returns:

0 on success, -1 on failure

This is a mode for using block cipher (AES in this case) for authentication. OMAC1 was standardized with the name CMAC by NIST in a Special Publication (SP) 800-38B.

15.90 src/crypto/aes-unwrap.c File Reference

AES key unwrap (128-bit KEK, RFC3394). `#include "includes.h"`

`#include "common.h"`

`#include "aes.h"`

`#include "aes_wrap.h"`

Functions

- int aes_unwrap (const u8 *kek, int n, const u8 *cipher, u8 *plain)
  Unwrap key with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).

15.90.1 Detailed Description

AES key unwrap (128-bit KEK, RFC3394).

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.90.2 Function Documentation

15.90.2.1 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 plaintext key in 64-bit units; e.g., 2 = 128-bit = 16 bytes

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

plain Plaintext key, n * 64 bits

Returns:

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

15.91 src/crypto/aes-wrap.c File Reference

AES Key Wrap Algorithm (128-bit KEK) (RFC3394). `#include "includes.h"`

`#include "common.h"`

`#include "aes.h"`

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).

15.91.1 Detailed Description

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

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.91.2 Function Documentation

15.91.2.1 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 16-octet Key encryption key (KEK)

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

plain Plaintext key to be wrapped, n * 64 bits

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

Returns:

0 on success, -1 on failure

15.92 src/crypto/aes.h File Reference

AES functions.

Defines

- #define **AES_BLOCK_SIZE** 16

Functions

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

15.92.1 Detailed Description

AES functions.

Copyright

Copyright (c) 2003-2006, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.93 src/crypto/aes_i.h File Reference

AES (Rijndael) cipher. `#include "aes.h"`

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 **GETU32**(pt)
- #define **PUTU32**(ct, st)
- #define **AES_PRIV_SIZE** (4 * 44)

Functions

- void rijndaelKeySetupEnc (u32 rk[ ], const u8 cipherKey[ ])

Variables

- const u32 **Te0** [256]
- const u32 **Te1** [256]
- const u32 **Te2** [256]
- const u32 **Te3** [256]
- const u32 **Te4** [256]
- const u32 **Td0** [256]

- const u32 **Td1** [256]
- const u32 **Td2** [256]
- const u32 **Td3** [256]
- const u32 **Td4** [256]
- const u32 **rcon** [10]
- const u8 **Td4s** [256]
- const u8 **rcons** [10]

15.93.1 Detailed Description

AES (Rijndael) cipher.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.93.2 Define Documentation

15.93.2.1 #define GETU32(pt)

Value:

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

15.93.2.2 #define PUTU32(ct, st)

Value:

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

15.93.3 Function Documentation

15.93.3.1 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.

15.94 src/crypto/aes_wrap.h File Reference

AES-based functions.

Functions

- int __must_check aes_wrap (const u8 *kek, int n, const u8 *plain, u8 *cipher)
  Wrap keys with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).
- int __must_check aes_unwrap (const u8 *kek, int n, const u8 *cipher, u8 *plain)
  Unwrap key with AES Key Wrap Algorithm (128-bit KEK) (RFC3394).
- int __must_check omac1_aes_128_vector (const u8 *key, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  One-Key CBC MAC (OMAC1) hash with AES-128.
- int __must_check omac1_aes_128 (const u8 *key, const u8 *data, size_t data_len, u8 *mac)
  One-Key CBC MAC (OMAC1) hash with AES-128 (aka AES-CMAC).
- int __must_check aes_128_encrypt_block (const u8 *key, const u8 *in, u8 *out)
  Perform one AES 128-bit block operation.
- int __must_check aes_128_ctr_encrypt (const u8 *key, const u8 *nonce, u8 *data, size_t data_len)
  AES-128 CTR mode encryption.
- int __must_check 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 __must_check 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 __must_check aes_128_cbc_encrypt (const u8 *key, const u8 *iv, u8 *data, size_t data_len)
  AES-128 CBC encryption.
- int __must_check aes_128_cbc_decrypt (const u8 *key, const u8 *iv, u8 *data, size_t data_len)
  AES-128 CBC decryption.

15.94.1 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) 2003-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.94.2 Function Documentation

15.94.2.1 int __must_check 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

15.94.2.2 int __must_check 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

15.94.2.3 int __must_check 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

15.94.2.4 int __must_check 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

15.94.2.5 int __must_check 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

15.94.2.6 int __must_check 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

15.94.2.7 int __must_check 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 plaintext key in 64-bit units; e.g., 2 = 128-bit = 16 bytes

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

plain Plaintext key, n * 64 bits

Returns:

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

15.94.2.8 int __must_check 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 16-octet Key encryption key (KEK)

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

plain Plaintext key to be wrapped, n * 64 bits

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

Returns:

0 on success, -1 on failure

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

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

Parameters:

key 128-bit key for the hash operation

data Data buffer for which a MAC is determined

data_len Length of data buffer in bytes

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

Returns:

0 on success, -1 on failure

This is a mode for using block cipher (AES in this case) for authentication. OMAC1 was standardized with the name CMAC by NIST in a Special Publication (SP) 800-38B.

15.94.2.10 int __must_check omac1_aes_128_vector (const u8 * *key*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

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

Parameters:

key 128-bit key for the hash operation

num_elem Number of elements in the data vector

addr Pointers to the data areas

len Lengths of the data blocks

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

Returns:

0 on success, -1 on failure

This is a mode for using block cipher (AES in this case) for authentication. OMAC1 was standardized with the name CMAC by NIST in a Special Publication (SP) 800-38B.

15.95 src/crypto/crypto.h File Reference

WPA Supplicant / wrapper functions for crypto libraries.

Defines

- #define **md5_vector_non_fips_allow** md5_vector

Enumerations

- enum **crypto_hash_alg** { **CRYPTO_HASH_ALG_MD5**, **CRYPTO_HASH_ALG_SHA1**, **CRYPTO_HASH_ALG_HMAC_MD5**, **CRYPTO_HASH_ALG_HMAC_SHA1** }
- enum **crypto_cipher_alg** {

  CRYPTO_CIPHER_NULL = 0, **CRYPTO_CIPHER_ALG_AES**, **CRYPTO_CIPHER_ALG_-3DES**, **CRYPTO_CIPHER_ALG_DES**,

  CRYPTO_CIPHER_ALG_RC2, **CRYPTO_CIPHER_ALG_RC4** }

Functions

- int md4_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  MD4 hash for data vector.

- int md5_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  MD5 hash for data vector.

- int sha1_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  SHA-1 hash for data vector.

- int __must_check fips186_2_prf (const u8 *seed, size_t seed_len, u8 *x, size_t xlen)

  NIST FIPS Publication 186-2 change notice 1 PRF.

- int sha256_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  SHA256 hash for data vector.

- void des_encrypt (const u8 *clear, const u8 *key, u8 *cypher)

  Encrypt one block with DES.

- void * aes_encrypt_init (const u8 *key, size_t len)

  Initialize AES for encryption.

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

  Encrypt one AES block.

- void aes_encrypt_deinit (void *ctx)

  Deinitialize AES encryption.

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

  Initialize AES for decryption.

- void aes_decrypt (void *ctx, const u8 *crypt, u8 *plain)
  Decrypt one AES block.
- void aes_decrypt_deinit (void *ctx)
  Deinitialize AES decryption.
- struct crypto_hash * crypto_hash_init (enum crypto_hash_alg alg, const u8 *key, size_t key_len)
  Initialize hash/HMAC function.
- void crypto_hash_update (struct crypto_hash *ctx, const u8 *data, size_t len)
  Add data to hash calculation.
- int crypto_hash_finish (struct crypto_hash *ctx, u8 *hash, size_t *len)
  Complete hash calculation.
- struct crypto_cipher * crypto_cipher_init (enum crypto_cipher_alg alg, const u8 *iv, const u8 *key, size_t key_len)
  Initialize block/stream cipher function.
- int __must_check crypto_cipher_encrypt (struct crypto_cipher *ctx, const u8 *plain, u8 *crypt, size_t len)
  Cipher encrypt.
- int __must_check crypto_cipher_decrypt (struct crypto_cipher *ctx, const u8 *crypt, u8 *plain, size_t len)
  Cipher decrypt.
- void crypto_cipher_deinit (struct crypto_cipher *ctx)
  Free cipher context.
- struct crypto_public_key * crypto_public_key_import (const u8 *key, size_t len)
  Import an RSA public key.
- struct crypto_private_key * crypto_private_key_import (const u8 *key, size_t len, const char *passwd)
  Import an RSA private key.
- struct crypto_public_key * crypto_public_key_from_cert (const u8 *buf, size_t len)
  Import an RSA public key from a certificate.
- int __must_check crypto_public_key_encrypt_pkcs1_v15 (struct crypto_public_key *key, const u8 *in, size_t inlen, u8 *out, size_t *outlen)
  Public key encryption (PKCS #1 v1.5).
- int __must_check crypto_private_key_decrypt_pkcs1_v15 (struct crypto_private_key *key, const u8 *in, size_t inlen, u8 *out, size_t *outlen)
  Private key decryption (PKCS #1 v1.5).
- int __must_check crypto_private_key_sign_pkcs1 (struct crypto_private_key *key, const u8 *in, size_t inlen, u8 *out, size_t *outlen)

Sign with private key (PKCS #1).

- void crypto_public_key_free (struct crypto_public_key *key)

  Free public key.

- void crypto_private_key_free (struct crypto_private_key *key)

  Free private key.

- int __must_check crypto_public_key_decrypt_pkcs1 (struct crypto_public_key *key, const u8 *crypt, size_t crypt_len, u8 *plain, size_t *plain_len)

  Decrypt PKCS #1 signature.

- int __must_check crypto_global_init (void)

  Initialize crypto wrapper.

- void crypto_global_deinit (void)

  Deinitialize crypto wrapper.

- int __must_check crypto_mod_exp (const u8 *base, size_t base_len, const u8 *power, size_t power_len, const u8 *modulus, size_t modulus_len, u8 *result, size_t *result_len)

  Modular exponentiation of large integers.

- int 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.

15.95.1 Detailed Description

WPA Supplicant / wrapper functions for crypto libraries.

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

15.95.2 Function Documentation

15.95.2.1 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)

15.95.2.2 void aes_decrypt_deinit (void * *ctx*)

Deinitialize AES decryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.95.2.3 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

15.95.2.4 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)

15.95.2.5 void aes_encrypt_deinit (void * *ctx*)

Deinitialize AES encryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.95.2.6 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

15.95.2.7 int __must_check crypto_cipher_decrypt (struct crypto_cipher * *ctx*, const u8 * *crypt*, u8 * *plain*, size_t *len*)

Cipher decrypt.

Parameters:

ctx Context pointer from crypto_cipher_init()

crypt Ciphertext to decrypt

plain Resulting plaintext

len Length of the cipher text

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.8 void crypto_cipher_deinit (struct crypto_cipher * *ctx*)

Free cipher context.

Parameters:

ctx Context pointer from crypto_cipher_init()

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.9 int __must_check crypto_cipher_encrypt (struct crypto_cipher * *ctx*, const u8 * *plain*, u8 * *crypt*, size_t *len*)

Cipher encrypt.

Parameters:

ctx Context pointer from crypto_cipher_init()

plain Plaintext to cipher

crypt Resulting ciphertext

len Length of the plaintext

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.10 struct crypto_cipher* crypto_cipher_init (enum crypto_cipher_alg *alg*, const u8 * *iv*, const u8 * *key*, size_t *key_len*) [read]

Initialize block/stream cipher function.

Parameters:

alg Cipher algorithm

iv Initialization vector for block ciphers or NULL for stream ciphers

key Cipher key

key_len Length of key in bytes

Returns:

Pointer to cipher context to use with other cipher functions or NULL on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.11 void crypto_global_deinit (void)

Deinitialize crypto wrapper. This function is only used with internal TLSv1 implementation (CONFIG_-TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.12 int __must_check crypto_global_init (void)

Initialize crypto wrapper. This function is only used with internal TLSv1 implementation (CONFIG_-TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.13 int crypto_hash_finish (struct crypto_hash * *ctx*, u8 * *hash*, size_t * *len*)

Complete hash calculation.

Parameters:

ctx Context pointer from crypto_hash_init()

hash Buffer for hash value or NULL if caller is just freeing the hash context

len Pointer to length of the buffer or NULL if caller is just freeing the hash context; on return, this is set to the actual length of the hash value

Returns:

0 on success, -1 if buffer is too small (len set to needed length), or -2 on other failures (including failed crypto_hash_update() operations)

This function calculates the hash value and frees the context buffer that was used for hash calculation.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.14 struct crypto_hash* crypto_hash_init (enum crypto_hash_alg *alg*, const u8 * *key*, size_t *key_len*) [read]

Initialize hash/HMAC function.

Parameters:

alg Hash algorithm

key Key for keyed hash (e.g., HMAC) or NULL if not needed

key_len Length of the key in bytes

Returns:

Pointer to hash context to use with other hash functions or NULL on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.15 void crypto_hash_update (struct crypto_hash * *ctx*, const u8 * *data*, size_t *len*)

Add data to hash calculation.

Parameters:

ctx Context pointer from crypto_hash_init()

data Data buffer to add

len Length of the buffer

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.16 int __must_check crypto_mod_exp (const u8 * *base*, size_t *base_len*, const u8 * *power*, size_t *power_len*, const u8 * *modulus*, size_t *modulus_len*, u8 * *result*, size_t * *result_len*)

Modular exponentiation of large integers.

Parameters:

base Base integer (big endian byte array)

base_len Length of base integer in bytes

power Power integer (big endian byte array)

power_len Length of power integer in bytes

modulus Modulus integer (big endian byte array)

modulus_len Length of modulus integer in bytes

result Buffer for the result

result_len Result length (max buffer size on input, real len on output)

Returns:

0 on success, -1 on failure

This function calculates result = base ^ power mod modulus. modules_len is used as the maximum size of modulus buffer. It is set to the used size on success.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.17 int __must_check crypto_private_key_decrypt_pkcs1_v15 (struct crypto_private_key * *key*, const u8 * *in*, size_t *inlen*, u8 * *out*, size_t * *outlen*)

Private key decryption (PKCS #1 v1.5).

Parameters:

key Private key

in Encrypted buffer

inlen Length of encrypted buffer in bytes

out Output buffer for encrypted data

outlen Length of output buffer in bytes; set to used length on success

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.18 void crypto_private_key_free (struct crypto_private_key * *key*)

Free private key.

Parameters:

key Private key from crypto_private_key_import()

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.19 struct crypto_private_key* crypto_private_key_import (const u8 * *key*, size_t *len*, const char * *passwd*) [read]

Import an RSA private key.

Parameters:

key Key buffer (DER encoded RSA private key)

len Key buffer length in bytes

passwd Key encryption password or NULL if key is not encrypted

Returns:

Pointer to the private key or NULL on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.20 int __must_check crypto_private_key_sign_pkcs1 (struct crypto_private_key * *key*, const u8 * *in*, size_t *inlen*, u8 * *out*, size_t * *outlen*)

Sign with private key (PKCS #1).

Parameters:

key Private key from crypto_private_key_import()

in Plaintext buffer

inlen Length of plaintext buffer in bytes

out Output buffer for encrypted (signed) data

outlen Length of output buffer in bytes; set to used length on success

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.21 int __must_check crypto_public_key_decrypt_pkcs1 (struct crypto_public_key * *key*, const u8 * *crypt*, size_t *crypt_len*, u8 * *plain*, size_t * *plain_len*)

Decrypt PKCS #1 signature.

Parameters:

key Public key

crypt Encrypted signature data (using the private key)

crypt_len Encrypted signature data length

plain Buffer for plaintext (at least crypt_len bytes)

plain_len Plaintext length (max buffer size on input, real len on output);

Returns:

0 on success, -1 on failure

15.95.2.22 int __must_check crypto_public_key_encrypt_pkcs1_v15 (struct crypto_public_key * *key*, const u8 * *in*, size_t *inlen*, u8 * *out*, size_t * *outlen*)

Public key encryption (PKCS #1 v1.5).

Parameters:

key Public key

in Plaintext buffer

inlen Length of plaintext buffer in bytes

out Output buffer for encrypted data

outlen Length of output buffer in bytes; set to used length on success

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.23 void crypto_public_key_free (struct crypto_public_key * *key*)

Free public key.

Parameters:

key Public key

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.24 struct crypto_public_key* crypto_public_key_from_cert (const u8 * *buf*, size_t *len*) [read]

Import an RSA public key from a certificate.

Parameters:

buf DER encoded X.509 certificate

len Certificate buffer length in bytes

Returns:

Pointer to public key or NULL on failure

This function can just return NULL if the crypto library does not support X.509 parsing. In that case, internal code will be used to parse the certificate and public key is imported using crypto_public_key_import().

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.25 struct crypto_public_key* crypto_public_key_import (const u8 * *key*, size_t *len*) [read]

Import an RSA public key.

Parameters:

key Key buffer (DER encoded RSA public key)

len Key buffer length in bytes

Returns:

Pointer to the public key or NULL on failure

This function can just return NULL if the crypto library supports X.509 parsing. In that case, crypto_public_key_from_cert() is used to import the public key from a certificate.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.95.2.26 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)

15.95.2.27 int __must_check fips186_2_prf (const u8 * *seed*, size_t *seed_len*, u8 * *x*, size_t *xlen*)

NIST FIPS Publication 186-2 change notice 1 PRF.

Parameters:

seed Seed/key for the PRF

seed_len Seed length in bytes

x Buffer for PRF output

xlen Output length in bytes

Returns:

0 on success, -1 on failure

This function implements random number generation specified in NIST FIPS Publication 186-2 for EAP-SIM. This PRF uses a function that is similar to SHA-1, but has different message padding.

15.95.2.28 int 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

Returns:

0 on success, -1 on failure

15.95.2.29 int 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

Returns:

0 on success, -1 on failure

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

Returns:

0 on success, -1 of failure

15.95.2.30 int 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

Returns:

0 on success, -1 on failure

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.

15.95.2.31 int 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

Returns:

0 on success, -1 on failure

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

Returns:

0 on success, -1 of failure

15.95.2.32 int sha256_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

SHA256 hash for data vector.

Parameters:

num_elem Number of elements in the data vector

addr Pointers to the data areas

len Lengths of the data blocks

mac Buffer for the hash

Returns:

0 on success, -1 on failure

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

Returns:

0 on success, -1 of failure

15.96 src/crypto/crypto_cryptoapi.c File Reference

WPA Supplicant / Crypto wrapper for Microsoft CryptoAPI. `#include "includes.h"`

```
#include <windows.h>
#include <wincrypt.h>
#include "common.h"
#include "crypto.h"
```

Defines

- #define **MS_ENH_RSA_AES_PROV** "Microsoft Enhanced RSA and AES Cryptographic Provider (Prototype)"
- #define **CALG_HMAC** (ALG_CLASS_HASH | ALG_TYPE_ANY | ALG_SID_HMAC)

Functions

- int **cryptoapi_hash_vector** (ALG_ID alg, size_t hash_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
- int 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.

15.96.1 Detailed Description

WPA Supplicant / Crypto wrapper for Microsoft CryptoAPI.

Copyright

Copyright (c) 2005-2006, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.96.2 Function Documentation

15.96.2.1 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)

15.96.2.2 int 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

Returns:

0 on success, -1 on failure

15.97 src/crypto/crypto_gnutls.c File Reference

WPA Supplicant / wrapper functions for libgcrypt. `#include "includes.h"`

```
#include <gcrypt.h>
#include "common.h"
#include "crypto.h"
```

Data Structures

- struct crypto_cipher

Functions

- int 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.
- int md5_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  MD5 hash for data vector.
- int sha1_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  SHA-1 hash for data vector.
- 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.
- int crypto_mod_exp (const u8 *base, size_t base_len, const u8 *power, size_t power_len, const u8 *modulus, size_t modulus_len, u8 *result, size_t *result_len)
  Modular exponentiation of large integers.

- struct crypto_cipher * crypto_cipher_init (enum crypto_cipher_alg alg, const u8 *iv, const u8 *key, size_t key_len)

  Initialize block/stream cipher function.

- int crypto_cipher_encrypt (struct crypto_cipher *ctx, const u8 *plain, u8 *crypt, size_t len)

  Cipher encrypt.

- int crypto_cipher_decrypt (struct crypto_cipher *ctx, const u8 *crypt, u8 *plain, size_t len)

  Cipher decrypt.

- void crypto_cipher_deinit (struct crypto_cipher *ctx)

  Free cipher context.

15.97.1 Detailed Description

WPA Supplicant / wrapper functions for libgcrypt.

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.97.2 Function Documentation

15.97.2.1 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)

15.97.2.2 void aes_decrypt_deinit (void * *ctx*)

Deinitialize AES decryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.97.2.3 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

15.97.2.4 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)

15.97.2.5 void aes_encrypt_deinit (void * *ctx*)

Deinitialize AES encryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.97.2.6 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

15.97.2.7 int crypto_cipher_decrypt (struct crypto_cipher * *ctx*, const u8 * *crypt*, u8 * *plain*, size_t *len*)

Cipher decrypt.

Parameters:

ctx Context pointer from crypto_cipher_init()

crypt Ciphertext to decrypt

plain Resulting plaintext

len Length of the cipher text

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.97.2.8 void crypto_cipher_deinit (struct crypto_cipher * *ctx*)

Free cipher context.

Parameters:

ctx Context pointer from crypto_cipher_init()

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.97.2.9 int crypto_cipher_encrypt (struct crypto_cipher * *ctx*, const u8 * *plain*, u8 * *crypt*, size_t *len*)

Cipher encrypt.

Parameters:

ctx Context pointer from crypto_cipher_init()

plain Plaintext to cipher

crypt Resulting ciphertext

len Length of the plaintext

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.97.2.10 struct crypto_cipher* crypto_cipher_init (enum crypto_cipher_alg *alg*, const u8 * *iv*, const u8 * *key*, size_t *key_len*) [read]

Initialize block/stream cipher function.

Parameters:

alg Cipher algorithm

iv Initialization vector for block ciphers or NULL for stream ciphers

key Cipher key

key_len Length of key in bytes

Returns:

Pointer to cipher context to use with other cipher functions or NULL on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.97.2.11 int crypto_mod_exp (const u8 * *base*, size_t *base_len*, const u8 * *power*, size_t *power_len*, const u8 * *modulus*, size_t *modulus_len*, u8 * *result*, size_t * *result_len*)

Modular exponentiation of large integers.

Parameters:

base Base integer (big endian byte array)

base_len Length of base integer in bytes

power Power integer (big endian byte array)

power_len Length of power integer in bytes

modulus Modulus integer (big endian byte array)

modulus_len Length of modulus integer in bytes

result Buffer for the result

result_len Result length (max buffer size on input, real len on output)

Returns:

0 on success, -1 on failure

This function calculates result = base ^ power mod modulus. modules_len is used as the maximum size of modulus buffer. It is set to the used size on success.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.97.2.12 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)

15.97.2.13 int 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

Returns:

0 on success, -1 on failure

15.97.2.14 int 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

Returns:

0 on success, -1 on failure

15.97.2.15 int 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

Returns:

0 on success, -1 on failure

15.98 src/crypto/crypto_internal.c File Reference

WPA Supplicant / Crypto wrapper for internal crypto implementation. `#include "includes.h"`

```
#include "common.h"
#include "crypto.h"
#include "md5.h"
#include "sha1.h"
#include "aes.h"
#include "tls/rsa.h"
#include "tls/bignum.h"
#include "tls/pkcs1.h"
#include "tls/pkcs8.h"
#include "sha1_i.h"
#include "md5_i.h"
#include "des_i.h"
```

15.98.1 Detailed Description

WPA Supplicant / Crypto wrapper for internal crypto implementation.

Copyright

15.99 src/crypto/crypto_libtomcrypt.c File Reference

WPA Supplicant / Crypto wrapper for LibTomCrypt (for internal TLSv1). `#include "includes.h"`

```
#include <tomcrypt.h>
#include "common.h"
#include "crypto.h"
```

Defines

- #define **mp_init_multi** ltc_init_multi
- #define **mp_clear_multi** ltc_deinit_multi
- #define **mp_unsigned_bin_size**(a) ltc_mp.unsigned_size(a)
- #define **mp_to_unsigned_bin**(a, b) ltc_mp.unsigned_write(a, b)
- #define **mp_read_unsigned_bin**(a, b, c) ltc_mp.unsigned_read(a, b, c)
- #define **mp_exptmod**(a, b, c, d) ltc_mp.exptmod(a,b,c,d)

Functions

- int 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.

15.99.1 Detailed Description

WPA Supplicant / Crypto wrapper for LibTomCrypt (for internal TLSv1).

Copyright

Copyright (c) 2005-2006, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.99.2 Function Documentation

15.99.2.1 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)

15.99.2.2 int 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

Returns:

0 on success, -1 on failure

15.100 src/crypto/crypto_none.c File Reference

WPA Supplicant / Empty template functions for crypto wrapper. `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

Functions

- int 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.

15.100.1 Detailed Description

WPA Supplicant / Empty template functions for crypto wrapper.

Copyright

Copyright (c) 2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.100.2 Function Documentation

15.100.2.1 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)

15.100.2.2 int 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

Returns:

0 on success, -1 on failure

15.101 src/crypto/crypto_nss.c File Reference

Crypto wrapper functions for NSS. `#include "includes.h"`

```
#include <nspr/prtypes.h>
#include <nspr/plarenas.h>
#include <nspr/plhash.h>
#include <nspr/prtime.h>
#include <nspr/prinrval.h>
#include <nspr/prclist.h>
#include <nspr/prlock.h>
#include <nss/sechash.h>
#include <nss/pk11pub.h>
#include "common.h"
#include "crypto.h"
```

Data Structures

- struct crypto_cipher

Functions

- void des_encrypt (const u8 *clear, const u8 *key, u8 *cypher)
  Encrypt one block with DES.
- int 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.
- int md5_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  MD5 hash for data vector.
- int sha1_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  SHA-1 hash for data vector.
- int sha256_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  SHA256 hash for data vector.
- 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.
- int crypto_mod_exp (const u8 *base, size_t base_len, const u8 *power, size_t power_len, const u8 *modulus, size_t modulus_len, u8 *result, size_t *result_len)
  Modular exponentiation of large integers.
- struct crypto_cipher * crypto_cipher_init (enum crypto_cipher_alg alg, const u8 *iv, const u8 *key, size_t key_len)
  Initialize block/stream cipher function.
- int crypto_cipher_encrypt (struct crypto_cipher *ctx, const u8 *plain, u8 *crypt, size_t len)
  Cipher encrypt.
- int crypto_cipher_decrypt (struct crypto_cipher *ctx, const u8 *crypt, u8 *plain, size_t len)
  Cipher decrypt.
- void crypto_cipher_deinit (struct crypto_cipher *ctx)
  Free cipher context.

15.101.1 Detailed Description

Crypto wrapper functions for NSS.

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.101.2 Function Documentation

15.101.2.1 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)

15.101.2.2 void aes_decrypt_deinit (void * *ctx*)

Deinitialize AES decryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.101.2.3 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

15.101.2.4 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)

15.101.2.5 void aes_encrypt_deinit (void * *ctx*)

Deinitialize AES encryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.101.2.6 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

15.101.2.7 int crypto_cipher_decrypt (struct crypto_cipher * *ctx*, const u8 * *crypt*, u8 * *plain*, size_t *len*)

Cipher decrypt.

Parameters:

ctx Context pointer from crypto_cipher_init()

crypt Ciphertext to decrypt

plain Resulting plaintext

len Length of the cipher text

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.101.2.8 void crypto_cipher_deinit (struct crypto_cipher * *ctx*)

Free cipher context.

Parameters:

ctx Context pointer from crypto_cipher_init()

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.101.2.9 int crypto_cipher_encrypt (struct crypto_cipher * *ctx*, const u8 * *plain*, u8 * *crypt*, size_t *len*)

Cipher encrypt.

Parameters:

ctx Context pointer from crypto_cipher_init()

plain Plaintext to cipher

crypt Resulting ciphertext

len Length of the plaintext

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.101.2.10 struct crypto_cipher* crypto_cipher_init (enum crypto_cipher_alg *alg*, const u8 * *iv*, const u8 * *key*, size_t *key_len*) [read]

Initialize block/stream cipher function.

Parameters:

alg Cipher algorithm

iv Initialization vector for block ciphers or NULL for stream ciphers

key Cipher key

key_len Length of key in bytes

Returns:

Pointer to cipher context to use with other cipher functions or NULL on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.101.2.11 int crypto_mod_exp (const u8 * *base*, size_t *base_len*, const u8 * *power*, size_t *power_len*, const u8 * *modulus*, size_t *modulus_len*, u8 * *result*, size_t * *result_len*)

Modular exponentiation of large integers.

Parameters:

base Base integer (big endian byte array)

base_len Length of base integer in bytes

power Power integer (big endian byte array)

power_len Length of power integer in bytes

modulus Modulus integer (big endian byte array)

modulus_len Length of modulus integer in bytes

result Buffer for the result

result_len Result length (max buffer size on input, real len on output)

Returns:

0 on success, -1 on failure

This function calculates result = base ^ power mod modulus. modules_len is used as the maximum size of modulus buffer. It is set to the used size on success.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.101.2.12 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)

15.101.2.13 int 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

Returns:

0 on success, -1 on failure

15.101.2.14 int 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

Returns:

0 on success, -1 on failure

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.

15.101.2.15 int 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

Returns:

0 on success, -1 on failure

15.101.2.16 int sha256_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

SHA256 hash for data vector.

Parameters:

num_elem Number of elements in the data vector

addr Pointers to the data areas

len Lengths of the data blocks

mac Buffer for the hash

Returns:

0 on success, -1 on failure

15.102 src/crypto/crypto_openssl.c File Reference

WPA Supplicant / wrapper functions for libcrypto. `#include "includes.h"`

```
#include <openssl/opensslv.h>
#include <openssl/err.h>
#include <openssl/des.h>
#include <openssl/aes.h>
#include <openssl/bn.h>
#include <openssl/evp.h>
#include <openssl/dh.h>
#include "common.h"
#include "wpabuf.h"
#include "dh_group5.h"
#include "crypto.h"
```

Data Structures

- struct crypto_cipher

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))
- #define **NO_SHA256_WRAPPER**

Functions

- int 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.

- int 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.

- int md5_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  MD5 hash for data vector.

- int sha1_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  SHA-1 hash for data vector.

- 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.
- int crypto_mod_exp (const u8 *base, size_t base_len, const u8 *power, size_t power_len, const u8 *modulus, size_t modulus_len, u8 *result, size_t *result_len)
  Modular exponentiation of large integers.
- struct crypto_cipher * crypto_cipher_init (enum crypto_cipher_alg alg, const u8 *iv, const u8 *key, size_t key_len)
  Initialize block/stream cipher function.
- int crypto_cipher_encrypt (struct crypto_cipher *ctx, const u8 *plain, u8 *crypt, size_t len)
  Cipher encrypt.
- int crypto_cipher_decrypt (struct crypto_cipher *ctx, const u8 *crypt, u8 *plain, size_t len)
  Cipher decrypt.
- void crypto_cipher_deinit (struct crypto_cipher *ctx)
  Free cipher context.
- void * **dh5_init** (struct wpabuf **priv, struct wpabuf **publ)
- struct wpabuf * **dh5_derive_shared** (void *ctx, const struct wpabuf *peer_public, const struct wpabuf *own_private)
- void **dh5_free** (void *ctx)

15.102.1 Detailed Description

WPA Supplicant / wrapper functions for libcrypto.

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.102.2 Function Documentation

15.102.2.1 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)

15.102.2.2 void aes_decrypt_deinit (void * *ctx*)

Deinitialize AES decryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.102.2.3 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

15.102.2.4 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)

15.102.2.5 void aes_encrypt_deinit (void * *ctx*)

Deinitialize AES encryption.

Parameters:

ctx Context pointer from aes_encrypt_init()

15.102.2.6 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

15.102.2.7 int crypto_cipher_decrypt (struct crypto_cipher * *ctx*, const u8 * *crypt*, u8 * *plain*, size_t *len*)

Cipher decrypt.

Parameters:

ctx Context pointer from crypto_cipher_init()

crypt Ciphertext to decrypt

plain Resulting plaintext

len Length of the cipher text

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.102.2.8 void crypto_cipher_deinit (struct crypto_cipher * *ctx*)

Free cipher context.

Parameters:

ctx Context pointer from crypto_cipher_init()

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.102.2.9 int crypto_cipher_encrypt (struct crypto_cipher * *ctx*, const u8 * *plain*, u8 * *crypt*, size_t *len*)

Cipher encrypt.

Parameters:

ctx Context pointer from crypto_cipher_init()

plain Plaintext to cipher

crypt Resulting ciphertext

len Length of the plaintext

Returns:

0 on success, -1 on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.102.2.10 struct crypto_cipher* crypto_cipher_init (enum crypto_cipher_alg *alg*, const u8 * *iv*, const u8 * *key*, size_t *key_len*) [read]

Initialize block/stream cipher function.

Parameters:

alg Cipher algorithm

iv Initialization vector for block ciphers or NULL for stream ciphers

key Cipher key

key_len Length of key in bytes

Returns:

Pointer to cipher context to use with other cipher functions or NULL on failure

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.102.2.11 int crypto_mod_exp (const u8 * *base*, size_t *base_len*, const u8 * *power*, size_t *power_len*, const u8 * *modulus*, size_t *modulus_len*, u8 * *result*, size_t * *result_len*)

Modular exponentiation of large integers.

Parameters:

base Base integer (big endian byte array)

base_len Length of base integer in bytes

power Power integer (big endian byte array)

power_len Length of power integer in bytes

modulus Modulus integer (big endian byte array)

modulus_len Length of modulus integer in bytes

result Buffer for the result

result_len Result length (max buffer size on input, real len on output)

Returns:

0 on success, -1 on failure

This function calculates result = base ^ power mod modulus. modules_len is used as the maximum size of modulus buffer. It is set to the used size on success.

This function is only used with internal TLSv1 implementation (CONFIG_TLS=internal). If that is not used, the crypto wrapper does not need to implement this.

15.102.2.12 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)

15.102.2.13 int 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

Returns:

0 on success, -1 on failure

15.102.2.14 int 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

Returns:

0 on success, -1 on failure

15.102.2.15 int 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

Returns:

0 on success, -1 on failure

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.

15.102.2.16 int 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

Returns:

0 on success, -1 on failure

15.103 src/crypto/des-internal.c File Reference

DES and 3DES-EDE ciphers. `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

`#include "des_i.h"`

Defines

- #define ROLc(x, y)
- #define **RORc**(x, y)

Functions

- void des_encrypt (const u8 *clear, const u8 *key, u8 *cypher)

  Encrypt one block with DES.

- void **des_key_setup** (const u8 *key, u32 *ek, u32 *dk)
- void **des_block_encrypt** (const u8 *plain, const u32 *ek, u8 *crypt)
- void **des_block_decrypt** (const u8 *crypt, const u32 *dk, u8 *plain)
- void **des3_key_setup** (const u8 *key, struct des3_key_s *dkey)
- void **des3_encrypt** (const u8 *plain, const struct des3_key_s *key, u8 *crypt)
- void **des3_decrypt** (const u8 *crypt, const struct des3_key_s *key, u8 *plain)

15.103.1 Detailed Description

DES and 3DES-EDE ciphers. Modifications to LibTomCrypt implementation:

Copyright

Copyright (c) 2006-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.103.2 Define Documentation

15.103.2.1 #define ROLc(x, y)

Value:

```
((((unsigned long) (x) << (unsigned long) ((y) & 31)) | \
          (((unsigned long) (x) & 0xFFFFFFFFUL) >> \
           (unsigned long) (32 - ((y) & 31)))) & 0xFFFFFFFFUL)
```

DES code submitted by Dobes Vandermeer

15.103.2.2 #define RORc(x, y)

Value:

```
(((((unsigned long) (x) & 0xFFFFFFFFUL) >> \
           (unsigned long) ((y) & 31)) | \
          ((unsigned long) (x) << (unsigned long) (32 - ((y) & 31)))) & \
         0xFFFFFFFFUL)
```

15.103.3 Function Documentation

15.103.3.1 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)

15.104 src/crypto/des_i.h File Reference

DES and 3DES-EDE ciphers.

Data Structures

- struct des3_key_s

Functions

- void **des_key_setup** (const u8 *key, u32 *ek, u32 *dk)
- void **des_block_encrypt** (const u8 *plain, const u32 *ek, u8 *crypt)
- void **des_block_decrypt** (const u8 *crypt, const u32 *dk, u8 *plain)
- void **des3_key_setup** (const u8 *key, struct des3_key_s *dkey)
- void **des3_encrypt** (const u8 *plain, const struct des3_key_s *key, u8 *crypt)
- void **des3_decrypt** (const u8 *crypt, const struct des3_key_s *key, u8 *plain)

15.104.1 Detailed Description

DES and 3DES-EDE ciphers.

Copyright

15.105 src/crypto/dh_group5.c File Reference

Diffie-Hellman group 5 operations. `#include "includes.h"`

`#include "common.h"`

`#include "dh_groups.h"`

`#include "dh_group5.h"`

Functions

- void * **dh5_init** (struct wpabuf **priv, struct wpabuf **publ)
- struct wpabuf * **dh5_derive_shared** (void *ctx, const struct wpabuf *peer_public, const struct wpabuf *own_private)
- void **dh5_free** (void *ctx)

15.105.1 Detailed Description

Diffie-Hellman group 5 operations.

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

15.106 src/crypto/dh_group5.h File Reference

Diffie-Hellman group 5 operations.

Functions

- void * **dh5_init** (struct wpabuf **priv, struct wpabuf **publ)
- struct wpabuf * **dh5_derive_shared** (void *ctx, const struct wpabuf *peer_public, const struct wpabuf *own_private)
- void **dh5_free** (void *ctx)

15.106.1 Detailed Description

Diffie-Hellman group 5 operations.

Copyright

15.107 src/crypto/dh_groups.c File Reference

Diffie-Hellman groups. `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

`#include "dh_groups.h"`

Defines

- #define **DH_GROUP**(id)
- #define **NUM_DH_GROUPS** (sizeof(dh_groups) / sizeof(dh_groups[0]))

Functions

- struct dh_group * **dh_groups_get** (int id)
- struct wpabuf * dh_init (const struct dh_group *dh, struct wpabuf **priv)

  Initialize Diffie-Hellman handshake.

- struct wpabuf * dh_derive_shared (const struct wpabuf *peer_public, const struct wpabuf *own_private, const struct dh_group *dh)

  Derive shared Diffie-Hellman key.

15.107.1 Detailed Description

Diffie-Hellman groups.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.107.2 Define Documentation

15.107.2.1 #define DH_GROUP(id)

Value:

```
{ id, dh_group ## id ## _generator, sizeof(dh_group ## id ## _generator), \
dh_group ## id ## _prime, sizeof(dh_group ## id ## _prime) }
```

15.107.3 Function Documentation

15.107.3.1 struct wpabuf* dh_derive_shared (const struct wpabuf * *peer_public*, const struct wpabuf * *own_private*, const struct dh_group * *dh*) [read]

Derive shared Diffie-Hellman key.

Parameters:

peer_public Diffie-Hellman public value from peer

own_private Diffie-Hellman private key from dh_init()

dh Selected Diffie-Hellman group

Returns:

Diffie-Hellman shared key

15.107.3.2 struct wpabuf* dh_init (const struct dh_group * *dh*, struct wpabuf ** *priv*) [read]

Initialize Diffie-Hellman handshake.

Parameters:

dh Selected Diffie-Hellman group

priv Pointer for returning Diffie-Hellman private key

Returns:

Diffie-Hellman public value

15.108 src/crypto/dh_groups.h File Reference

Diffie-Hellman groups.

Data Structures

- struct dh_group

Functions

- struct dh_group * **dh_groups_get** (int id)
- struct wpabuf * dh_init (const struct dh_group *dh, struct wpabuf **priv)

  Initialize Diffie-Hellman handshake.

- struct wpabuf * dh_derive_shared (const struct wpabuf *peer_public, const struct wpabuf *own_private, const struct dh_group *dh)

  Derive shared Diffie-Hellman key.

15.108.1 Detailed Description

Diffie-Hellman groups.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.108.2 Function Documentation

15.108.2.1 struct wpabuf* dh_derive_shared (const struct wpabuf * *peer_public*, const struct wpabuf * *own_private*, const struct dh_group * *dh*) [read]

Derive shared Diffie-Hellman key.

Parameters:

peer_public Diffie-Hellman public value from peer

own_private Diffie-Hellman private key from dh_init()

dh Selected Diffie-Hellman group

Returns:

Diffie-Hellman shared key

15.108.2.2 struct wpabuf* dh_init (const struct dh_group * *dh*, struct wpabuf ** *priv*) [read]

Initialize Diffie-Hellman handshake.

Parameters:

dh Selected Diffie-Hellman group

priv Pointer for returning Diffie-Hellman private key

Returns:

Diffie-Hellman public value

15.109 src/crypto/fips_prf_gnutls.c File Reference

FIPS 186-2 PRF for libgcrypt. `#include "includes.h"`

`#include <gcrypt.h>`

`#include "common.h"`

`#include "crypto.h"`

Functions

- int fips186_2_prf (const u8 *seed, size_t seed_len, u8 *x, size_t xlen)
  NIST FIPS Publication 186-2 change notice 1 PRF.

15.109.1 Detailed Description

FIPS 186-2 PRF for libgcrypt.

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.109.2 Function Documentation

15.109.2.1 int fips186_2_prf (const u8 * *seed*, size_t *seed_len*, u8 * *x*, size_t *xlen*)

NIST FIPS Publication 186-2 change notice 1 PRF.

Parameters:

seed Seed/key for the PRF

seed_len Seed length in bytes

x Buffer for PRF output

xlen Output length in bytes

Returns:

0 on success, -1 on failure

This function implements random number generation specified in NIST FIPS Publication 186-2 for EAP-SIM. This PRF uses a function that is similar to SHA-1, but has different message padding.

15.110 src/crypto/fips_prf_internal.c File Reference

FIPS 186-2 PRF for internal crypto implementation. `#include "includes.h"`

`#include "common.h"`

`#include "sha1.h"`

`#include "sha1_i.h"`

`#include "crypto.h"`

Functions

- int fips186_2_prf (const u8 *seed, size_t seed_len, u8 *x, size_t xlen)
  NIST FIPS Publication 186-2 change notice 1 PRF.

15.110.1 Detailed Description

FIPS 186-2 PRF for internal crypto implementation.

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.110.2 Function Documentation

15.110.2.1 int fips186_2_prf (const u8 * *seed*, size_t *seed_len*, u8 * *x*, size_t *xlen*)

NIST FIPS Publication 186-2 change notice 1 PRF.

Parameters:

seed Seed/key for the PRF

seed_len Seed length in bytes

x Buffer for PRF output

xlen Output length in bytes

Returns:

0 on success, -1 on failure

This function implements random number generation specified in NIST FIPS Publication 186-2 for EAP-SIM. This PRF uses a function that is similar to SHA-1, but has different message padding.

15.111 src/crypto/fips_prf_nss.c File Reference

FIPS 186-2 PRF for NSS. `#include "includes.h"`

`#include <openssl/sha.h>`

`#include "common.h"`

`#include "crypto.h"`

Functions

- int fips186_2_prf (const u8 *seed, size_t seed_len, u8 *x, size_t xlen)
  NIST FIPS Publication 186-2 change notice 1 PRF.

15.111.1 Detailed Description

FIPS 186-2 PRF for NSS.

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.111.2 Function Documentation

15.111.2.1 int fips186_2_prf (const u8 * *seed*, size_t *seed_len*, u8 * *x*, size_t *xlen*)

NIST FIPS Publication 186-2 change notice 1 PRF.

Parameters:

seed Seed/key for the PRF

seed_len Seed length in bytes

x Buffer for PRF output

xlen Output length in bytes

Returns:

0 on success, -1 on failure

This function implements random number generation specified in NIST FIPS Publication 186-2 for EAP-SIM. This PRF uses a function that is similar to SHA-1, but has different message padding.

15.112 src/crypto/fips_prf_openssl.c File Reference

FIPS 186-2 PRF for libcrypto. `#include "includes.h"`

`#include <openssl/sha.h>`

`#include "common.h"`

`#include "crypto.h"`

Functions

- int fips186_2_prf (const u8 *seed, size_t seed_len, u8 *x, size_t xlen)
  NIST FIPS Publication 186-2 change notice 1 PRF.

15.112.1 Detailed Description

FIPS 186-2 PRF for libcrypto.

Copyright

Copyright (c) 2004-2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.112.2 Function Documentation

15.112.2.1 int fips186_2_prf (const u8 * *seed*, size_t *seed_len*, u8 * *x*, size_t *xlen*)

NIST FIPS Publication 186-2 change notice 1 PRF.

Parameters:

seed Seed/key for the PRF

seed_len Seed length in bytes

x Buffer for PRF output

xlen Output length in bytes

Returns:

0 on success, -1 on failure

This function implements random number generation specified in NIST FIPS Publication 186-2 for EAP-SIM. This PRF uses a function that is similar to SHA-1, but has different message padding.

15.113 src/crypto/md4-internal.c File Reference

MD4 hash implementation. `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

Data Structures

- struct MD4Context

Defines

- #define **MD4_BLOCK_LENGTH** 64
- #define **MD4_DIGEST_LENGTH** 16
- #define **MD4_DIGEST_STRING_LENGTH** (MD4_DIGEST_LENGTH * 2 + 1)
- #define **PUT_64BIT_LE**(cp, value)
- #define **PUT_32BIT_LE**(cp, value)
- #define **F1**(x, y, z) (z ^ (x & (y ^ z)))
- #define **F2**(x, y, z) ((x & y) | (x & z) | (y & z))
- #define **F3**(x, y, z) (x ^ y ^ z)
- #define **MD4STEP**(f, w, x, y, z, data, s) (w += f(x, y, z) + data, w = w<<s | w>>(32-s))

Typedefs

- typedef struct MD4Context **MD4_CTX**

Functions

- int md4_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  MD4 hash for data vector.

15.113.1 Detailed Description

MD4 hash implementation.

Copyright

Copyright (c) 2006, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.113.2 Define Documentation

15.113.2.1 #define PUT_32BIT_LE(cp, value)

Value:

```
do {                                    \
        (cp)[3] = (value) >> 24;                                        \
        (cp)[2] = (value) >> 16;                                        \
        (cp)[1] = (value) >> 8;                                         \
        (cp)[0] = (value); } while (0)
```

15.113.2.2 #define PUT_64BIT_LE(cp, value)

Value:

```
do {                                    \
        (cp)[7] = (value) >> 56;                                        \
        (cp)[6] = (value) >> 48;                                        \
        (cp)[5] = (value) >> 40;                                        \
        (cp)[4] = (value) >> 32;                                        \
        (cp)[3] = (value) >> 24;                                        \
        (cp)[2] = (value) >> 16;                                        \
        (cp)[1] = (value) >> 8;                                         \
        (cp)[0] = (value); } while (0)
```

15.113.3 Function Documentation

15.113.3.1 int 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

Returns:

0 on success, -1 on failure

15.114 src/crypto/md5-internal.c File Reference

MD5 hash implementation and interface functions. `#include "includes.h"`

```
#include "common.h"
#include "md5.h"
#include "md5_i.h"
#include "crypto.h"
```

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 struct MD5Context **MD5_CTX**

Functions

- int md5_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  MD5 hash for data vector.

- void **MD5Init** (struct MD5Context *ctx)
- void **MD5Update** (struct MD5Context *ctx, unsigned char const *buf, unsigned len)
- void **MD5Final** (unsigned char digest[16], struct MD5Context *ctx)

15.114.1 Detailed Description

MD5 hash implementation and interface functions.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.114.2 Function Documentation

15.114.2.1 int 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

Returns:

0 on success, -1 of failure

15.115 src/crypto/md5-non-fips.c File Reference

MD5 hash implementation and interface functions (non-FIPS allowed cases). `#include "includes.h"`

`#include "common.h"`

`#include "md5.h"`

`#include "crypto.h"`

Functions

- int hmac_md5_vector_non_fips_allow (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).
- int hmac_md5_non_fips_allow (const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *mac)
  HMAC-MD5 over data buffer (RFC 2104).

15.115.1 Detailed Description

MD5 hash implementation and interface functions (non-FIPS allowed cases).

Copyright

Copyright (c) 2003-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.115.2 Function Documentation

15.115.2.1 int hmac_md5_non_fips_allow (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)

Returns:

0 on success, -1 on failure

15.115.2.2 int hmac_md5_vector_non_fips_allow (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)

Returns:

0 on success, -1 on failure

15.116 src/crypto/md5.c File Reference

MD5 hash implementation and interface functions. `#include "includes.h"`

`#include "common.h"`

`#include "md5.h"`

`#include "crypto.h"`

Functions

- int 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).

- int 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).

15.116.1 Detailed Description

MD5 hash implementation and interface functions.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.116.2 Function Documentation

15.116.2.1 int 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)

Returns:

0 on success, -1 on failure

15.116.2.2 int 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)

Returns:

0 on success, -1 on failure

15.117 src/crypto/md5.h File Reference

MD5 hash implementation and interface functions.

Defines

- #define **MD5_MAC_LEN** 16
- #define **hmac_md5_vector_non_fips_allow** hmac_md5_vector
- #define **hmac_md5_non_fips_allow** hmac_md5

Functions

- int 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).

- int 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).

15.117.1 Detailed Description

MD5 hash implementation and interface functions.

Copyright

Copyright (c) 2003-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.117.2 Function Documentation

15.117.2.1 int 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)

Returns:

0 on success, -1 on failure

15.117.2.2 int 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)

Returns:

0 on success, -1 on failure

15.118 src/crypto/md5_i.h File Reference

MD5 internal definitions.

Data Structures

- struct MD5Context

Functions

- void **MD5Init** (struct MD5Context *context)
- void **MD5Update** (struct MD5Context *context, unsigned char const *buf, unsigned len)
- void **MD5Final** (unsigned char digest[16], struct MD5Context *context)

15.118.1 Detailed Description

MD5 internal definitions.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.119 src/crypto/ms_funcs.c File Reference

WPA Supplicant / shared MSCHAPV2 helper functions / RFC 2433 / RFC 2759. `#include "includes.h"`

`#include "common.h"`

`#include "sha1.h"`

`#include "ms_funcs.h"`

`#include "crypto.h"`

Defines

- #define **PWBLOCK_LEN** 516

Functions

- int nt_password_hash (const u8 *password, size_t password_len, u8 *password_hash)

  NtPasswordHash() - RFC 2759, Sect. 8.3.

- int 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.

- int 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.

- int generate_nt_response_pwhash (const u8 *auth_challenge, const u8 *peer_challenge, const u8 *username, size_t username_len, const u8 *password_hash, u8 *response)

  GenerateNTResponse() - RFC 2759, Sect. 8.1.

- int generate_authenticator_response_pwhash (const u8 *password_hash, const u8 *peer_challenge, const u8 *auth_challenge, const u8 *username, size_t username_len, const u8 *nt_response, u8 *response)

  GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

- int 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.

- int nt_challenge_response (const u8 *challenge, const u8 *password, size_t password_len, u8 *response)

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

- int get_master_key (const u8 *password_hash_hash, const u8 *nt_response, u8 *master_key)

  GetMasterKey() - RFC 3079, Sect. 3.4.

- int get_asymetric_start_key (const u8 *master_key, u8 *session_key, size_t session_key_len, int is_send, int is_server)

  GetAsymetricStartKey() - RFC 3079, Sect. 3.4.

- int encrypt_pw_block_with_password_hash (const u8 *password, size_t password_len, const u8 *password_hash, u8 *pw_block)

  EncryptPwBlockWithPasswordHash() - RFC 2759, Sect. 8.10.

- int 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 nt_password_hash_encrypted_with_block (const u8 *password_hash, const u8 *block, u8 *cypher)

  NtPasswordHashEncryptedWithBlock() - RFC 2759, Sect 8.13.

- int 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.

15.119.1 Detailed Description

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

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

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

See README and COPYING for more details.

15.119.2 Function Documentation

15.119.2.1 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)

15.119.2.2 int encrypt_pw_block_with_password_hash (const u8 * *password*, size_t *password_len*, const u8 * *password_hash*, u8 * *pw_block*)

EncryptPwBlockWithPasswordHash() - RFC 2759, Sect. 8.10.

Parameters:

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

password_len Length of password

password_hash 16-octet PasswordHash (IN)

pw_block 516-byte PwBlock (OUT)

Returns:

0 on success, -1 on failure

15.119.2.3 int generate_authenticator_response (const u8 * *password*, size_t *password_len*, const u8 * *peer_challenge*, const u8 * *auth_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *nt_response*, u8 * *response*)

GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

Parameters:

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

password_len Length of password

nt_response 24-octet NT-Response (IN)

peer_challenge 16-octet PeerChallenge (IN)

auth_challenge 16-octet AuthenticatorChallenge (IN)

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

username_len Length of username

response 20-octet AuthenticatorResponse (OUT) (note: this value is usually encoded as a 42-octet ASCII string (S=hexdump_of_response)

Returns:

0 on success, -1 on failure

15.119.2.4 int generate_authenticator_response_pwhash (const u8 * *password_hash*, const u8 * *peer_challenge*, const u8 * *auth_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *nt_response*, u8 * *response*)

GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

Parameters:

password_hash 16-octet PasswordHash (IN)

nt_response 24-octet NT-Response (IN)

peer_challenge 16-octet PeerChallenge (IN)

auth_challenge 16-octet AuthenticatorChallenge (IN)

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

username_len Length of username

response 20-octet AuthenticatorResponse (OUT) (note: this value is usually encoded as a 42-octet ASCII string (S=hexdump_of_response)

Returns:

0 on success, -1 on failure

15.119.2.5 int 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)

Returns:

0 on success, -1 on failure

15.119.2.6 int generate_nt_response_pwhash (const u8 * *auth_challenge*, const u8 * *peer_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *password_hash*, u8 * *response*)

GenerateNTResponse() - RFC 2759, Sect. 8.1.

Parameters:

auth_challenge 16-octet AuthenticatorChallenge (IN)

peer_challenge 16-octet PeerChallenge (IN)

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

username_len Length of username

password_hash 16-octet PasswordHash (IN)

response 24-octet Response (OUT)

Returns:

0 on success, -1 on failure

15.119.2.7 int get_asymetric_start_key (const u8 * *master_key*, u8 * *session_key*, size_t *session_key_len*, int *is_send*, int *is_server*)

GetAsymetricStartKey() - RFC 3079, Sect. 3.4.

Parameters:

master_key 16-octet MasterKey (IN)

session_key 8-to-16 octet SessionKey (OUT)

session_key_len SessionKeyLength (Length of session_key) (IN)

is_send IsSend (IN, BOOLEAN)

is_server IsServer (IN, BOOLEAN)

Returns:

0 on success, -1 on failure

15.119.2.8 int 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)

Returns:

0 on success, -1 on failure

15.119.2.9 int 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)

Returns:

0 on success, -1 on failure

15.119.2.10 int 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)

Returns:

0 on success, -1 on failure

15.119.2.11 int 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)

Returns:

0 on success, -1 on failure

15.119.2.12 int 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)

Returns:

0 on success, -1 on failure

15.119.2.13 void nt_password_hash_encrypted_with_block (const u8 * *password_hash*, const u8 * *block*, u8 * *cypher*)

NtPasswordHashEncryptedWithBlock() - RFC 2759, Sect 8.13.

Parameters:

password_hash 16-octer PasswordHash (IN)

block 16-octet Block (IN)

cypher 16-octer Cypher (OUT)

15.119.2.14 int 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_hash 16-octet EncryptedPasswordHash (OUT)

Returns:

0 on success, -1 on failure

15.120 src/crypto/ms_funcs.h File Reference

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

Functions

- int 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.

- int generate_nt_response_pwhash (const u8 *auth_challenge, const u8 *peer_challenge, const u8 *username, size_t username_len, const u8 *password_hash, u8 *response)

  GenerateNTResponse() - RFC 2759, Sect. 8.1.

- int 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.

- int generate_authenticator_response_pwhash (const u8 *password_hash, const u8 *peer_challenge, const u8 *auth_challenge, const u8 *username, size_t username_len, const u8 *nt_response, u8 *response)

  GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

- int 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.

- int nt_password_hash (const u8 *password, size_t password_len, u8 *password_hash)

  NtPasswordHash() - RFC 2759, Sect. 8.3.

- int hash_nt_password_hash (const u8 *password_hash, u8 *password_hash_hash)

  HashNtPasswordHash() - RFC 2759, Sect. 8.4.

- int get_master_key (const u8 *password_hash_hash, const u8 *nt_response, u8 *master_key)

  GetMasterKey() - RFC 3079, Sect. 3.4.

- int get_asymetric_start_key (const u8 *master_key, u8 *session_key, size_t session_key_len, int is_send, int is_server)

  GetAsymetricStartKey() - RFC 3079, Sect. 3.4.

- int __must_check encrypt_pw_block_with_password_hash (const u8 *password, size_t password_len, const u8 *password_hash, u8 *pw_block)

  EncryptPwBlockWithPasswordHash() - RFC 2759, Sect. 8.10.

- int __must_check 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 nt_password_hash_encrypted_with_block (const u8 *password_hash, const u8 *block, u8 *cypher)

  NtPasswordHashEncryptedWithBlock() - RFC 2759, Sect 8.13.

- int 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.

15.120.1 Detailed Description

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

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

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

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.120.2 Function Documentation

15.120.2.1 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)

15.120.2.2 int __must_check encrypt_pw_block_with_password_hash (const u8 * *password*, size_t *password_len*, const u8 * *password_hash*, u8 * *pw_block*)

EncryptPwBlockWithPasswordHash() - RFC 2759, Sect. 8.10.

Parameters:

password 0-to-256-unicode-char Password (IN; ASCII)

password_len Length of password

password_hash 16-octet PasswordHash (IN)

pw_block 516-byte PwBlock (OUT)

Returns:

0 on success, -1 on failure

15.120.2.3 int generate_authenticator_response (const u8 * *password*, size_t *password_len*, const u8 * *peer_challenge*, const u8 * *auth_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *nt_response*, u8 * *response*)

GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

Parameters:

password 0-to-256-unicode-char Password (IN; ASCII)

password_len Length of password

nt_response 24-octet NT-Response (IN)

peer_challenge 16-octet PeerChallenge (IN)

auth_challenge 16-octet AuthenticatorChallenge (IN)

username 0-to-256-char UserName (IN)

username_len Length of username

response 20-octet AuthenticatorResponse (OUT) (note: this value is usually encoded as a 42-octet ASCII string (S=hexdump_of_response)

Returns:

0 on success, -1 on failure

15.120.2.4 int generate_authenticator_response_pwhash (const u8 * *password_hash*, const u8 * *peer_challenge*, const u8 * *auth_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *nt_response*, u8 * *response*)

GenerateAuthenticatorResponse() - RFC 2759, Sect. 8.7.

Parameters:

password_hash 16-octet PasswordHash (IN)

nt_response 24-octet NT-Response (IN)

peer_challenge 16-octet PeerChallenge (IN)

auth_challenge 16-octet AuthenticatorChallenge (IN)

username 0-to-256-char UserName (IN)

username_len Length of username

response 20-octet AuthenticatorResponse (OUT) (note: this value is usually encoded as a 42-octet ASCII string (S=hexdump_of_response)

Returns:

0 on success, -1 on failure

15.120.2.5 int 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)

Returns:

0 on success, -1 on failure

15.120.2.6 int generate_nt_response_pwhash (const u8 * *auth_challenge*, const u8 * *peer_challenge*, const u8 * *username*, size_t *username_len*, const u8 * *password_hash*, u8 * *response*)

GenerateNTResponse() - RFC 2759, Sect. 8.1.

Parameters:

auth_challenge 16-octet AuthenticatorChallenge (IN)

peer_challenge 16-octet PeerChallenge (IN)

username 0-to-256-char UserName (IN)

username_len Length of username

password_hash 16-octet PasswordHash (IN)

response 24-octet Response (OUT)

Returns:

0 on success, -1 on failure

15.120.2.7 int get_asymetric_start_key (const u8 * *master_key*, u8 * *session_key*, size_t *session_key_len*, int *is_send*, int *is_server*)

GetAsymetricStartKey() - RFC 3079, Sect. 3.4.

Parameters:

master_key 16-octet MasterKey (IN)

session_key 8-to-16 octet SessionKey (OUT)

session_key_len SessionKeyLength (Length of session_key) (IN)

is_send IsSend (IN, BOOLEAN)

is_server IsServer (IN, BOOLEAN)

Returns:

0 on success, -1 on failure

15.120.2.8 int 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)

Returns:

0 on success, -1 on failure

15.120.2.9 int 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)

Returns:

0 on success, -1 on failure

15.120.2.10 int __must_check 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)

Returns:

0 on success, -1 on failure

15.120.2.11 int 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)

Returns:

0 on success, -1 on failure

15.120.2.12 int 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)

Returns:

0 on success, -1 on failure

15.120.2.13 void nt_password_hash_encrypted_with_block (const u8 * *password_hash*, const u8 * *block*, u8 * *cypher*)

NtPasswordHashEncryptedWithBlock() - RFC 2759, Sect 8.13.

Parameters:

password_hash 16-octer PasswordHash (IN)

block 16-octet Block (IN)

cypher 16-octer Cypher (OUT)

15.120.2.14 int 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_hash 16-octet EncryptedPasswordHash (OUT)

Returns:

0 on success, -1 on failure

15.121 src/crypto/rc4.c File Reference

RC4 stream cipher. `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

Defines

- #define **S_SWAP**(a, b) do { u8 t = S[a]; S[a] = S[b]; S[b] = t; } while(0)

Functions

- int 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.

15.121.1 Detailed Description

RC4 stream cipher.

Copyright

Copyright (c) 2002-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.121.2 Function Documentation

15.121.2.1 int 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

Returns:

0 on success, -1 on failure

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.

15.122 src/crypto/sha1-internal.c File Reference

SHA1 hash implementation and interface functions. `#include "includes.h"`

```
#include "common.h"
#include "sha1.h"
#include "sha1_i.h"
#include "md5.h"
#include "crypto.h"
```

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)

Typedefs

- typedef struct SHA1Context **SHA1_CTX**

Functions

- void **SHA1Transform** (u32 state[5], const unsigned char buffer[64])
- int sha1_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  SHA-1 hash for data vector.

- void **SHA1Init** (SHA1_CTX *context)
- void **SHA1Update** (SHA1_CTX *context, const void *_data, u32 len)
- void **SHA1Final** (unsigned char digest[20], SHA1_CTX *context)

15.122.1 Detailed Description

SHA1 hash implementation and interface functions.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.122.2 Define Documentation

15.122.2.1 #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))
```

15.122.2.2 #define blk0(i)

Value:

```
(block->l[i] = (rol(block->l[i], 24) & 0xFF00FF00) | \
        (rol(block->l[i], 8) & 0x00FF00FF))
```

15.122.2.3 #define R0(v, w, x, y, z, i)

Value:

```
z += ((w & (x ^ y)) ^ y) + blk0(i) + 0x5A827999 + rol(v, 5); \
        w = rol(w, 30);
```

15.122.2.4 #define R1(v, w, x, y, z, i)

Value:

```
z += ((w & (x ^ y)) ^ y) + blk(i) + 0x5A827999 + rol(v, 5); \
        w = rol(w, 30);
```

15.122.2.5 #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);
```

15.122.2.6 #define R4(v, w, x, y, z, i)

Value:

```
z += (w ^ x ^ y) + blk(i) + 0xCA62C1D6 + rol(v, 5); \
        w=rol(w, 30);
```

15.122.3 Function Documentation

15.122.3.1 int 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

Returns:

0 on success, -1 of failure

15.123 src/crypto/sha1-pbkdf2.c File Reference

SHA1-based key derivation function (PBKDF2) for IEEE 802.11i. `#include "includes.h"`

`#include "common.h"`

`#include "sha1.h"`

`#include "md5.h"`

`#include "crypto.h"`

Functions

- int 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.

15.123.1 Detailed Description

SHA1-based key derivation function (PBKDF2) for IEEE 802.11i.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.123.2 Function Documentation

15.123.2.1 int 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

Returns:

0 on success, -1 of failure

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.

15.124 src/crypto/sha1-tlsprf.c File Reference

TLS PRF (SHA1 + MD5). `#include "includes.h"`

`#include "common.h"`

`#include "sha1.h"`

`#include "md5.h"`

`#include "crypto.h"`

Functions

- 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).

15.124.1 Detailed Description

TLS PRF (SHA1 + MD5).

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.124.2 Function Documentation

15.124.2.1 int tls_prf (const u8 * *secret*, size_t *secret_len*, const char * *label*, const u8 * *seed*, size_t *seed_len*, u8 * *out*, size_t *outlen*)

Pseudo-Random Function for TLS (TLS-PRF, RFC 2246).

Parameters:

secret Key for PRF

secret_len Length of the key in bytes

label A unique label for each purpose of the PRF

seed Seed value to bind into the key

seed_len Length of the seed

out Buffer for the generated pseudo-random key

outlen Number of bytes of key to generate

Returns:

0 on success, -1 on failure.

This function is used to derive new, cryptographically separate keys from a given key in TLS. This PRF is defined in RFC 2246, Chapter 5.

15.125 src/crypto/sha1-tprf.c File Reference

SHA1 T-PRF for EAP-FAST. `#include "includes.h"`

`#include "common.h"`

`#include "sha1.h"`

`#include "crypto.h"`

Functions

- int 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).

15.125.1 Detailed Description

SHA1 T-PRF for EAP-FAST.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.125.2 Function Documentation

15.125.2.1 int 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

Returns:

0 on success, -1 of failure

This function is used to derive new, cryptographically separate keys from a given key for EAP-FAST. T-PRF is defined in RFC 4851, Section 5.5.

15.126 src/crypto/sha1.c File Reference

SHA1 hash implementation and interface functions. `#include "includes.h"`

```
#include "common.h"
#include "sha1.h"
#include "crypto.h"
```

Functions

- int 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).

- int 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).

- int 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).

15.126.1 Detailed Description

SHA1 hash implementation and interface functions.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.126.2 Function Documentation

15.126.2.1 int 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)

Returns:

0 on success, -1 of failure

15.126.2.2 int 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)

Returns:

0 on success, -1 on failure

15.126.2.3 int 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

Returns:

0 on success, -1 of failure

This function is used to derive new, cryptographically separate keys from a given key (e.g., PMK in IEEE 802.11i).

15.127 src/crypto/sha1.h File Reference

SHA1 hash implementation and interface functions.

Defines

- #define **SHA1_MAC_LEN** 20

Functions

- int 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).

- int 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).

- int 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).

- int 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 __must_check 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).

- int 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.

15.127.1 Detailed Description

SHA1 hash implementation and interface functions.

Copyright

15.127.2 Function Documentation

15.127.2.1 int 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)

Returns:

0 on success, -1 of failure

15.127.2.2 int 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)

Returns:

0 on success, -1 on failure

15.127.2.3 int 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

Returns:

0 on success, -1 of failure

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.

15.127.2.4 int 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

Returns:

0 on success, -1 of failure

This function is used to derive new, cryptographically separate keys from a given key (e.g., PMK in IEEE 802.11i).

15.127.2.5 int 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

Returns:

0 on success, -1 of failure

This function is used to derive new, cryptographically separate keys from a given key for EAP-FAST. T-PRF is defined in RFC 4851, Section 5.5.

15.127.2.6 int __must_check tls_prf (const u8 * *secret*, size_t *secret_len*, const char * *label*, const u8 * *seed*, size_t *seed_len*, u8 * *out*, size_t *outlen*)

Pseudo-Random Function for TLS (TLS-PRF, RFC 2246).

Parameters:

secret Key for PRF

secret_len Length of the key in bytes

label A unique label for each purpose of the PRF

seed Seed value to bind into the key

seed_len Length of the seed

out Buffer for the generated pseudo-random key

outlen Number of bytes of key to generate

Returns:

0 on success, -1 on failure.

This function is used to derive new, cryptographically separate keys from a given key in TLS. This PRF is defined in RFC 2246, Chapter 5.

15.128 src/crypto/sha1_i.h File Reference

SHA1 internal definitions.

Data Structures

- struct SHA1Context

Functions

- void **SHA1Init** (struct SHA1Context *context)
- void **SHA1Update** (struct SHA1Context *context, const void *data, u32 len)
- void **SHA1Final** (unsigned char digest[20], struct SHA1Context *context)
- void **SHA1Transform** (u32 state[5], const unsigned char buffer[64])

15.128.1 Detailed Description

SHA1 internal definitions.

Copyright

15.129 src/crypto/sha256-internal.c File Reference

SHA-256 hash implementation and interface functions. `#include "includes.h"`

`#include "common.h"`

`#include "sha256.h"`

`#include "crypto.h"`

Data Structures

- struct sha256_state

Defines

- #define **RORc**(x, y)
- #define **Ch**(x, y, z) (z ^ (x & (y ^ z)))
- #define **Maj**(x, y, z) (((x | y) & z) | (x & y))
- #define **S**(x, n) RORc((x), (n))
- #define **R**(x, n) (((x)&0xFFFFFFFFUL)>>(n))
- #define **Sigma0**(x) (S(x, 2) ^ S(x, 13) ^ S(x, 22))
- #define **Sigma1**(x) (S(x, 6) ^ S(x, 11) ^ S(x, 25))
- #define **Gamma0**(x) (S(x, 7) ^ S(x, 18) ^ R(x, 3))
- #define **Gamma1**(x) (S(x, 17) ^ S(x, 19) ^ R(x, 10))
- #define **MIN**(x, y) (((x) < (y)) ? (x) : (y))
- #define **RND**(a, b, c, d, e, f, g, h, i)
- #define **block_size** 64

Functions

- int sha256_vector (size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  SHA256 hash for data vector.

15.129.1 Detailed Description

SHA-256 hash implementation and interface functions.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.129.2 Define Documentation

15.129.2.1 #define RND(a, b, c, d, e, f, g, h, i)

Value:

```
t0 = h + Sigma1(e) + Ch(e, f, g) + K[i] + W[i]; \
        t1 = Sigma0(a) + Maj(a, b, c);                  \
        d += t0;                                        \
        h  = t0 + t1;
```

15.129.2.2 #define RORc(x, y)

Value:

```
( ((((unsigned long) (x) & 0xFFFFFFFFUL) >> (unsigned long) ((y) & 31)) | \
   ((unsigned long) (x) << (unsigned long) (32 - ((y) & 31)))) & 0xFFFFFFFFUL)
```

15.129.3 Function Documentation

15.129.3.1 int sha256_vector (size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

SHA256 hash for data vector.

Parameters:

num_elem Number of elements in the data vector

addr Pointers to the data areas

len Lengths of the data blocks

mac Buffer for the hash

Returns:

0 on success, -1 of failure

15.130 src/crypto/sha256.c File Reference

SHA-256 hash implementation and interface functions. `#include "includes.h"`

`#include "common.h"`

`#include "sha256.h"`

`#include "crypto.h"`

Functions

- void hmac_sha256_vector (const u8 *key, size_t key_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)

  HMAC-SHA256 over data vector (RFC 2104).

- void hmac_sha256 (const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *mac)

  HMAC-SHA256 over data buffer (RFC 2104).

- void sha256_prf (const u8 *key, size_t key_len, const char *label, const u8 *data, size_t data_len, u8 *buf, size_t buf_len)

  SHA256-based Pseudo-Random Function (IEEE 802.11r, 8.5.1.5.2).

15.130.1 Detailed Description

SHA-256 hash implementation and interface functions.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.130.2 Function Documentation

15.130.2.1 void hmac_sha256 (const u8 * *key*, size_t *key_len*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

HMAC-SHA256 over data buffer (RFC 2104).

Parameters:

key Key for HMAC operations

key_len Length of the key in bytes

data Pointers to the data area

data_len Length of the data area

mac Buffer for the hash (20 bytes)

15.130.2.2 void hmac_sha256_vector (const u8 * *key*, size_t *key_len*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

HMAC-SHA256 over data vector (RFC 2104).

Parameters:

key Key for HMAC operations

key_len Length of the key in bytes

num_elem Number of elements in the data vector

addr Pointers to the data areas

len Lengths of the data blocks

mac Buffer for the hash (32 bytes)

15.130.2.3 void sha256_prf (const u8 * *key*, size_t *key_len*, const char * *label*, const u8 * *data*, size_t *data_len*, u8 * *buf*, size_t *buf_len*)

SHA256-based Pseudo-Random Function (IEEE 802.11r, 8.5.1.5.2).

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.

15.131 src/crypto/sha256.h File Reference

SHA256 hash implementation and interface functions.

Defines

- #define **SHA256_MAC_LEN** 32

Functions

- void hmac_sha256_vector (const u8 *key, size_t key_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *mac)
  HMAC-SHA256 over data vector (RFC 2104).
- void hmac_sha256 (const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *mac)
  HMAC-SHA256 over data buffer (RFC 2104).
- void sha256_prf (const u8 *key, size_t key_len, const char *label, const u8 *data, size_t data_len, u8 *buf, size_t buf_len)
  SHA256-based Pseudo-Random Function (IEEE 802.11r, 8.5.1.5.2).

15.131.1 Detailed Description

SHA256 hash implementation and interface functions.

Copyright

Copyright (c) 2003-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.131.2 Function Documentation

15.131.2.1 void hmac_sha256 (const u8 * *key*, size_t *key_len*, const u8 * *data*, size_t *data_len*, u8 * *mac*)

HMAC-SHA256 over data buffer (RFC 2104).

Parameters:

key Key for HMAC operations
key_len Length of the key in bytes
data Pointers to the data area
data_len Length of the data area
mac Buffer for the hash (20 bytes)

15.131.2.2 void hmac_sha256_vector (const u8 * *key*, size_t *key_len*, size_t *num_elem*, const u8 * *addr*[ ], const size_t * *len*, u8 * *mac*)

HMAC-SHA256 over data vector (RFC 2104).

Parameters:

key Key for HMAC operations

key_len Length of the key in bytes

num_elem Number of elements in the data vector

addr Pointers to the data areas

len Lengths of the data blocks

mac Buffer for the hash (32 bytes)

15.131.2.3 void sha256_prf (const u8 * *key*, size_t *key_len*, const char * *label*, const u8 * *data*, size_t *data_len*, u8 * *buf*, size_t *buf_len*)

SHA256-based Pseudo-Random Function (IEEE 802.11r, 8.5.1.5.2).

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.

15.132 src/crypto/tls.h File Reference

WPA Supplicant / SSL/TLS interface definition.

Data Structures

- struct tls_keys
- struct tls_config
- struct tls_connection_params

  Parameters for TLS connection.

Defines

- #define **TLS_CAPABILITY_IA** 0x0001

Typedefs

- typedef int(∗ **tls_session_ticket_cb**)(void ∗ctx, const u8 ∗ticket, size_t len, const u8 ∗client_random, const u8 ∗server_random, u8 ∗master_secret)

Enumerations

- enum { **TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED** = -3, **TLS_SET_PARAMS_-ENGINE_PRV_INIT_FAILED** = -2 }
- enum {

  TLS_CIPHER_NONE, **TLS_CIPHER_RC4_SHA**, **TLS_CIPHER_AES128_SHA**, **TLS_-CIPHER_RSA_DHE_AES128_SHA**,

  TLS_CIPHER_ANON_DH_AES128_SHA }

Functions

- void ∗ tls_init (const struct tls_config ∗conf)

  Initialize TLS library.

- void tls_deinit (void ∗tls_ctx)

  Deinitialize TLS library.

- int tls_get_errors (void ∗tls_ctx)

  Process pending errors.

- struct tls_connection ∗ tls_connection_init (void ∗tls_ctx)

  Initialize a new TLS connection.

- void tls_connection_deinit (void ∗tls_ctx, struct tls_connection ∗conn)

  Free TLS connection data.

- int tls_connection_established (void *tls_ctx, struct tls_connection *conn)

  Has the TLS connection been completed?

- int tls_connection_shutdown (void *tls_ctx, struct tls_connection *conn)

  Shutdown TLS connection.

- int __must_check tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  Set TLS connection parameters.

- int __must_check tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)

  Set TLS parameters for all TLS connection.

- int __must_check tls_global_set_verify (void *tls_ctx, int check_crl)

  Set global certificate verification options.

- int __must_check tls_connection_set_verify (void *tls_ctx, struct tls_connection *conn, int verify_peer)

  Set certificate verification options.

- int __must_check tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)

  Set TLS/IA parameters.

- int __must_check tls_connection_get_keys (void *tls_ctx, struct tls_connection *conn, struct tls_keys *keys)

  Get master key and random data from TLS connection.

- int __must_check tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  Use TLS-PRF to derive keying material.

- u8 * tls_connection_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  Process TLS handshake (client side).

- u8 * tls_connection_server_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  Process TLS handshake (server side).

- int __must_check 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 __must_check 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 __must_check tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  Configure acceptable cipher suites.

- int __must_check tls_get_cipher (void *tls_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  Get current cipher name.

- int __must_check tls_connection_enable_workaround (void *tls_ctx, struct tls_connection *conn)

  Enable TLS workaround options.

- int __must_check tls_connection_client_hello_ext (void *tls_ctx, struct tls_connection *conn, int ext_type, const u8 *data, size_t data_len)

  Set TLS extension for ClientHello.

- int tls_connection_get_failed (void *tls_ctx, struct tls_connection *conn)

  Get connection failure status.

- int tls_connection_get_read_alerts (void *tls_ctx, struct tls_connection *conn)

  Get connection read alert status.

- int tls_connection_get_write_alerts (void *tls_ctx, struct tls_connection *conn)

  Get connection write alert status.

- int tls_connection_get_keyblock_size (void *tls_ctx, struct tls_connection *conn)

  Get TLS key_block size.

- unsigned int tls_capabilities (void *tls_ctx)

  Get supported TLS capabilities.

- int __must_check tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)

  Send a TLS/IA PhaseFinished message.

- int __must_check tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)

  Has final phase been completed.

- int __must_check tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  Permute TLS/IA inner secret.

- int __must_check **tls_connection_set_session_ticket_cb** (void *tls_ctx, struct tls_connection *conn, tls_session_ticket_cb cb, void *ctx)

15.132.1 Detailed Description

WPA Supplicant / SSL/TLS interface definition.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.132.2 Function Documentation

15.132.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

15.132.2.2 int __must_check 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

15.132.2.3 int __must_check 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.

15.132.2.4 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

15.132.2.5 int __must_check 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.

15.132.2.6 int __must_check 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.

15.132.2.7 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if TLS connection has been completed, 0 if not.

15.132.2.8 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

15.132.2.9 int tls_connection_get_keyblock_size (void * *tls_ctx*, struct tls_connection * *conn*)

Get TLS key_block size.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Size of the key_block for the negotiated cipher suite or -1 on failure

15.132.2.10 int __must_check 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

15.132.2.11 int tls_connection_get_read_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection read alert status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Number of times a fatal read (remote end reported error) has happened during this connection.

15.132.2.12 int tls_connection_get_write_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection write alert status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Number of times a fatal write (locally detected error) has happened during this connection.

15.132.2.13 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

appl_data Pointer to application data pointer, or NULL if dropped

appl_data_len Pointer to variable that is set to appl_data length

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

15.132.2.14 int __must_check tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

15.132.2.15 int __must_check tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

key Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

key_len Length of session key material

Returns:

0 on success, -1 on failure

15.132.2.16 int __must_check tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

final 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

out_data Pointer to output buffer (encrypted TLS/IA data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

15.132.2.17 struct tls_connection* tls_connection_init (void * *tls_ctx*) [read]

Initialize a new TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Connection context data, conn for other function calls

15.132.2.18 int __must_check tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

15.132.2.19 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if current session used session resumption, 0 if not

15.132.2.20 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

Returns:

pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

15.132.2.21 int __must_check tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.132.2.22 int __must_check tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

tls_ia 1 = enable TLS/IA

Returns:

0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

15.132.2.23 int __must_check 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.

15.132.2.24 int __must_check 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

15.132.2.25 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

15.132.2.26 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

Parameters:

tls_ctx TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

15.132.2.27 int __must_check 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.

15.132.2.28 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

15.132.2.29 int __must_check tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

params Global TLS parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.132.2.30 int __must_check 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

15.132.2.31 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

Parameters:

conf Configuration data for TLS library

Returns:

Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

15.133 src/crypto/tls_gnutls.c File Reference

WPA Supplicant / SSL/TLS interface functions for openssl. `#include "includes.h"`

```
#include <gnutls/gnutls.h>
#include <gnutls/x509.h>
#include "common.h"
#include "tls.h"
```

Data Structures

- struct cipher_suite_st
- struct security_parameters_st
- struct gnutls_session_int
- struct tls_global
- struct tls_connection

Defines

- #define **TLS_RANDOM_SIZE** 32
- #define **TLS_MASTER_SIZE** 48
- #define **GNUTLS_INTERNAL_STRUCTURE_HACK**

Typedefs

- typedef u8 **uint8**
- typedef unsigned char **opaque**

Functions

- void * tls_init (const struct tls_config *conf)

  Initialize TLS library.

- void tls_deinit (void *ssl_ctx)

  Deinitialize TLS library.

- int tls_get_errors (void *ssl_ctx)

  Process pending errors.

- struct tls_connection * tls_connection_init (void *ssl_ctx)

  Initialize a new TLS connection.

- void tls_connection_deinit (void *ssl_ctx, struct tls_connection *conn)

  Free TLS connection data.

- int tls_connection_established (void *ssl_ctx, struct tls_connection *conn)

  Has the TLS connection been completed?

- int tls_connection_shutdown (void *ssl_ctx, struct tls_connection *conn)

  Shutdown TLS connection.

- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  Set TLS connection parameters.

- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)

  Set TLS parameters for all TLS connection.

- int tls_global_set_verify (void *ssl_ctx, int check_crl)

  Set global certificate verification options.

- int tls_connection_set_verify (void *ssl_ctx, struct tls_connection *conn, int verify_peer)

  Set certificate verification options.

- int tls_connection_get_keys (void *ssl_ctx, struct tls_connection *conn, struct tls_keys *keys)

  Get master key and random data from TLS connection.

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  Use TLS-PRF to derive keying material.

- u8 * tls_connection_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  Process TLS handshake (client side).

- u8 * tls_connection_server_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  Process TLS handshake (server side).

- int tls_connection_encrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Encrypt data into TLS tunnel.

- int tls_connection_decrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Decrypt data from TLS tunnel.

- int tls_connection_resumed (void *ssl_ctx, struct tls_connection *conn)

  Was session resumption used.

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  Configure acceptable cipher suites.

- int tls_get_cipher (void *ssl_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  Get current cipher name.

- int tls_connection_enable_workaround (void *ssl_ctx, struct tls_connection *conn)

  Enable TLS workaround options.

- int tls_connection_client_hello_ext (void *ssl_ctx, struct tls_connection *conn, int ext_type, const u8 *data, size_t data_len)
  Set TLS extension for ClientHello.
- int tls_connection_get_failed (void *ssl_ctx, struct tls_connection *conn)
  Get connection failure status.
- int tls_connection_get_read_alerts (void *ssl_ctx, struct tls_connection *conn)
  Get connection read alert status.
- int tls_connection_get_write_alerts (void *ssl_ctx, struct tls_connection *conn)
  Get connection write alert status.
- int tls_connection_get_keyblock_size (void *tls_ctx, struct tls_connection *conn)
  Get TLS key_block size.
- unsigned int tls_capabilities (void *tls_ctx)
  Get supported TLS capabilities.
- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)
  Set TLS/IA parameters.
- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)
  Send a TLS/IA PhaseFinished message.
- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)
  Has final phase been completed.
- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)
  Permute TLS/IA inner secret.

Variables

- int **wpa_debug_show_keys**

15.133.1 Detailed Description

WPA Supplicant / SSL/TLS interface functions for openssl.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.133.2 Function Documentation

15.133.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

15.133.2.2 int tls_connection_client_hello_ext (void * *tls_ctx*, struct tls_connection * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ext_type Extension type

data Extension payload (NULL to remove extension)

data_len Extension payload length

Returns:

0 on success, -1 on failure

15.133.2.3 int tls_connection_decrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Pointer to input buffer (encrypted TLS data)

in_len Input buffer length

out_data Pointer to output buffer (decrypted data from TLS tunnel)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

15.133.2.4 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

15.133.2.5 int tls_connection_enable_workaround (void * *tls_ctx*, struct tls_connection * *conn*)

Enable TLS workaround options.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

This function is used to enable connection-specific workaround options for buffer SSL/TLS implementations.

15.133.2.6 int tls_connection_encrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Pointer to plaintext data to be encrypted

in_len Input buffer length

out_data Pointer to output buffer (encrypted TLS data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

15.133.2.7 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if TLS connection has been completed, 0 if not.

15.133.2.8 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

15.133.2.9 int tls_connection_get_keyblock_size (void * *tls_ctx*, struct tls_connection * *conn*)

Get TLS key_block size.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Size of the key_block for the negotiated cipher suite or -1 on failure

15.133.2.10 int tls_connection_get_keys (void * *tls_ctx*, struct tls_connection * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

keys Structure of key/random data (filled on success)

Returns:

0 on success, -1 on failure

15.133.2.11 int tls_connection_get_read_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection read alert status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Number of times a fatal read (remote end reported error) has happened during this connection.

15.133.2.12 int tls_connection_get_write_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection write alert status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Number of times a fatal write (locally detected error) has happened during this connection.

15.133.2.13 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

appl_data Pointer to application data pointer, or NULL if dropped

appl_data_len Pointer to variable that is set to appl_data length

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server

is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

15.133.2.14 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

15.133.2.15 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

key Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

key_len Length of session key material

Returns:

0 on success, -1 on failure

15.133.2.16 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

final 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

out_data Pointer to output buffer (encrypted TLS/IA data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

15.133.2.17 struct tls_connection* tls_connection_init (void * *tls_ctx*) [read]

Initialize a new TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Connection context data, conn for other function calls

15.133.2.18 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

15.133.2.19 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if current session used session resumption, 0 if not

15.133.2.20 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

Returns:

pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

15.133.2.21 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.133.2.22 int tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

tls_ia 1 = enable TLS/IA

Returns:

0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

15.133.2.23 int tls_connection_set_params (void * *tls_ctx*, struct tls_connection * *conn*, const struct tls_connection_params * *params*)

Set TLS connection parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

params Connection parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.133.2.24 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

verify_peer 1 = verify peer certificate

Returns:

0 on success, -1 on failure

15.133.2.25 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

15.133.2.26 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

Parameters:

tls_ctx TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

15.133.2.27 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

buf Buffer for the cipher name

buflen buf size

Returns:

0 on success, -1 on failure

Get the name of the currently used cipher.

15.133.2.28 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

15.133.2.29 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

params Global TLS parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.133.2.30 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

check_crl 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

Returns:

0 on success, -1 on failure

15.133.2.31 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

Parameters:

conf Configuration data for TLS library

Returns:

Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

15.134 src/crypto/tls_internal.c File Reference

WPA Supplicant / TLS interface functions and an internal TLS implementation. `#include "includes.h"`

`#include "common.h"`

`#include "tls.h"`

`#include "tls/tlsv1_client.h"`

`#include "tls/tlsv1_server.h"`

Data Structures

- struct tls_global
- struct tls_connection

Functions

- void * tls_init (const struct tls_config *conf)
  Initialize TLS library.
- void tls_deinit (void *ssl_ctx)
  Deinitialize TLS library.
- int tls_get_errors (void *tls_ctx)
  Process pending errors.
- struct tls_connection * tls_connection_init (void *tls_ctx)
  Initialize a new TLS connection.
- void tls_connection_deinit (void *tls_ctx, struct tls_connection *conn)
  Free TLS connection data.
- int tls_connection_established (void *tls_ctx, struct tls_connection *conn)
  Has the TLS connection been completed?
- int tls_connection_shutdown (void *tls_ctx, struct tls_connection *conn)
  Shutdown TLS connection.
- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)
  Set TLS connection parameters.
- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)
  Set TLS parameters for all TLS connection.
- int tls_global_set_verify (void *tls_ctx, int check_crl)
  Set global certificate verification options.
- int tls_connection_set_verify (void *tls_ctx, struct tls_connection *conn, int verify_peer)

Set certificate verification options.

- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)

  Set TLS/IA parameters.

- int tls_connection_get_keys (void *tls_ctx, struct tls_connection *conn, struct tls_keys *keys)

  Get master key and random data from TLS connection.

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  Use TLS-PRF to derive keying material.

- u8 * tls_connection_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  Process TLS handshake (client side).

- u8 * tls_connection_server_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  Process TLS handshake (server side).

- int tls_connection_encrypt (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Encrypt data into TLS tunnel.

- int tls_connection_decrypt (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Decrypt data from TLS tunnel.

- int tls_connection_resumed (void *tls_ctx, struct tls_connection *conn)

  Was session resumption used.

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  Configure acceptable cipher suites.

- int tls_get_cipher (void *tls_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  Get current cipher name.

- int tls_connection_enable_workaround (void *tls_ctx, struct tls_connection *conn)

  Enable TLS workaround options.

- int tls_connection_client_hello_ext (void *tls_ctx, struct tls_connection *conn, int ext_type, const u8 *data, size_t data_len)

  Set TLS extension for ClientHello.

- int tls_connection_get_failed (void *tls_ctx, struct tls_connection *conn)

  Get connection failure status.

- int tls_connection_get_read_alerts (void *tls_ctx, struct tls_connection *conn)

  Get connection read alert status.

- int tls_connection_get_write_alerts (void *tls_ctx, struct tls_connection *conn)

  Get connection write alert status.

- int tls_connection_get_keyblock_size (void *tls_ctx, struct tls_connection *conn)

  Get TLS key_block size.

- unsigned int tls_capabilities (void *tls_ctx)

  Get supported TLS capabilities.

- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)

  Send a TLS/IA PhaseFinished message.

- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)

  Has final phase been completed.

- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  Permute TLS/IA inner secret.

- int **tls_connection_set_session_ticket_cb** (void *tls_ctx, struct tls_connection *conn, tls_session_ticket_cb cb, void *ctx)

15.134.1 Detailed Description

WPA Supplicant / TLS interface functions and an internal TLS implementation.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file interface functions for hostapd/wpa_supplicant to use the integrated TLSv1 implementation.

15.134.2 Function Documentation

15.134.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

15.134.2.2 int tls_connection_client_hello_ext (void * *tls_ctx*, struct tls_connection * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ext_type Extension type

data Extension payload (NULL to remove extension)

data_len Extension payload length

Returns:

0 on success, -1 on failure

15.134.2.3 int tls_connection_decrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Pointer to input buffer (encrypted TLS data)

in_len Input buffer length

out_data Pointer to output buffer (decrypted data from TLS tunnel)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

15.134.2.4 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

15.134.2.5 int tls_connection_enable_workaround (void * *tls_ctx*, struct tls_connection * *conn*)

Enable TLS workaround options.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

This function is used to enable connection-specific workaround options for buffer SSL/TLS implementations.

15.134.2.6 int tls_connection_encrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Pointer to plaintext data to be encrypted

in_len Input buffer length

out_data Pointer to output buffer (encrypted TLS data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

15.134.2.7 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if TLS connection has been completed, 0 if not.

15.134.2.8 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

15.134.2.9 int tls_connection_get_keyblock_size (void * *tls_ctx*, struct tls_connection * *conn*)

Get TLS key_block size.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Size of the key_block for the negotiated cipher suite or -1 on failure

15.134.2.10 int tls_connection_get_keys (void * *tls_ctx*, struct tls_connection * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

keys Structure of key/random data (filled on success)

Returns:

0 on success, -1 on failure

15.134.2.11 int tls_connection_get_read_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection read alert status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Number of times a fatal read (remote end reported error) has happened during this connection.

15.134.2.12 int tls_connection_get_write_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection write alert status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Number of times a fatal write (locally detected error) has happened during this connection.

15.134.2.13 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

appl_data Pointer to application data pointer, or NULL if dropped

appl_data_len Pointer to variable that is set to appl_data length

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

15.134.2.14 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

15.134.2.15 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

key Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

key_len Length of session key material

Returns:

0 on success, -1 on failure

15.134.2.16 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

final 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

out_data Pointer to output buffer (encrypted TLS/IA data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

15.134.2.17 struct tls_connection* tls_connection_init (void * *tls_ctx*) [read]

Initialize a new TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Connection context data, conn for other function calls

15.134.2.18 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

15.134.2.19 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if current session used session resumption, 0 if not

15.134.2.20 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

Returns:

pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

15.134.2.21 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.134.2.22 int tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

tls_ia 1 = enable TLS/IA

Returns:

0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

15.134.2.23 int tls_connection_set_params (void * *tls_ctx*, struct tls_connection * *conn*, const struct tls_connection_params * *params*)

Set TLS connection parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

params Connection parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.134.2.24 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

verify_peer 1 = verify peer certificate

Returns:

0 on success, -1 on failure

15.134.2.25 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

15.134.2.26 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

Parameters:

tls_ctx TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

15.134.2.27 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

buf Buffer for the cipher name

buflen buf size

Returns:

0 on success, -1 on failure

Get the name of the currently used cipher.

15.134.2.28 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

15.134.2.29 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

params Global TLS parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.134.2.30 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

check_crl 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

Returns:

0 on success, -1 on failure

15.134.2.31 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

Parameters:

conf Configuration data for TLS library

Returns:

Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

15.135 src/crypto/tls_none.c File Reference

WPA Supplicant / SSL/TLS interface functions for no TLS case. `#include "includes.h"`

`#include "common.h"`

`#include "tls.h"`

Functions

- void * tls_init (const struct tls_config *conf)
  Initialize TLS library.
- void tls_deinit (void *ssl_ctx)
  Deinitialize TLS library.

15.135.1 Detailed Description

WPA Supplicant / SSL/TLS interface functions for no TLS case.

Copyright

Copyright (c) 2004, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.135.2 Function Documentation

15.135.2.1 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.

15.135.2.2 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.

15.136 src/crypto/tls_nss.c File Reference

SSL/TLS interface functions for NSS. `#include "includes.h"`

```
#include <nspr/prtypes.h>
#include <nspr/plarenas.h>
#include <nspr/plhash.h>
#include <nspr/prio.h>
#include <nspr/prclist.h>
#include <nspr/prlock.h>
#include <nspr/prinit.h>
#include <nspr/prerror.h>
#include <nspr/prmem.h>
#include <nss/nss.h>
#include <nss/nssilckt.h>
#include <nss/ssl.h>
#include <nss/pk11func.h>
#include <nss/secerr.h>
#include "common.h"
#include "tls.h"
```

Data Structures

- struct tls_connection

Functions

- void * tls_init (const struct tls_config *conf)
  Initialize TLS library.
- void tls_deinit (void *ssl_ctx)
  Deinitialize TLS library.
- int tls_get_errors (void *tls_ctx)
  Process pending errors.
- struct tls_connection * tls_connection_init (void *tls_ctx)
  Initialize a new TLS connection.
- void tls_connection_deinit (void *tls_ctx, struct tls_connection *conn)
  Free TLS connection data.
- int tls_connection_established (void *tls_ctx, struct tls_connection *conn)
  Has the TLS connection been completed?

- int tls_connection_shutdown (void *tls_ctx, struct tls_connection *conn)

  Shutdown TLS connection.

- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  Set TLS connection parameters.

- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)

  Set TLS parameters for all TLS connection.

- int tls_global_set_verify (void *tls_ctx, int check_crl)

  Set global certificate verification options.

- int tls_connection_set_verify (void *tls_ctx, struct tls_connection *conn, int verify_peer)

  Set certificate verification options.

- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)

  Set TLS/IA parameters.

- int tls_connection_get_keys (void *tls_ctx, struct tls_connection *conn, struct tls_keys *keys)

  Get master key and random data from TLS connection.

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  Use TLS-PRF to derive keying material.

- u8 * tls_connection_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  Process TLS handshake (client side).

- u8 * tls_connection_server_handshake (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  Process TLS handshake (server side).

- int tls_connection_encrypt (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Encrypt data into TLS tunnel.

- int tls_connection_decrypt (void *tls_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Decrypt data from TLS tunnel.

- int tls_connection_resumed (void *tls_ctx, struct tls_connection *conn)

  Was session resumption used.

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  Configure acceptable cipher suites.

- int tls_get_cipher (void *tls_ctx, struct tls_connection *conn, char *buf, size_t buflen)

Get current cipher name.

- int tls_connection_enable_workaround (void *tls_ctx, struct tls_connection *conn)

  Enable TLS workaround options.

- int tls_connection_client_hello_ext (void *tls_ctx, struct tls_connection *conn, int ext_type, const u8 *data, size_t data_len)

  Set TLS extension for ClientHello.

- int tls_connection_get_failed (void *tls_ctx, struct tls_connection *conn)

  Get connection failure status.

- int tls_connection_get_read_alerts (void *tls_ctx, struct tls_connection *conn)

  Get connection read alert status.

- int tls_connection_get_write_alerts (void *tls_ctx, struct tls_connection *conn)

  Get connection write alert status.

- int tls_connection_get_keyblock_size (void *tls_ctx, struct tls_connection *conn)

  Get TLS key_block size.

- unsigned int tls_capabilities (void *tls_ctx)

  Get supported TLS capabilities.

- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)

  Send a TLS/IA PhaseFinished message.

- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)

  Has final phase been completed.

- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  Permute TLS/IA inner secret.

- int **tls_connection_set_session_ticket_cb** (void *tls_ctx, struct tls_connection *conn, tls_session_ticket_cb cb, void *ctx)

15.136.1 Detailed Description

SSL/TLS interface functions for NSS.

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.136.2 Function Documentation

15.136.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

15.136.2.2 int tls_connection_client_hello_ext (void * *tls_ctx*, struct tls_connection * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ext_type Extension type

data Extension payload (NULL to remove extension)

data_len Extension payload length

Returns:

0 on success, -1 on failure

15.136.2.3 int tls_connection_decrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Pointer to input buffer (encrypted TLS data)

in_len Input buffer length

out_data Pointer to output buffer (decrypted data from TLS tunnel)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

15.136.2.4 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

15.136.2.5 int tls_connection_enable_workaround (void * *tls_ctx*, struct tls_connection * *conn*)

Enable TLS workaround options.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

This function is used to enable connection-specific workaround options for buffer SSL/TLS implementations.

15.136.2.6 int tls_connection_encrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Pointer to plaintext data to be encrypted

in_len Input buffer length

out_data Pointer to output buffer (encrypted TLS data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

15.136.2.7 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if TLS connection has been completed, 0 if not.

15.136.2.8 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

15.136.2.9 int tls_connection_get_keyblock_size (void * *tls_ctx*, struct tls_connection * *conn*)

Get TLS key_block size.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Size of the key_block for the negotiated cipher suite or -1 on failure

15.136.2.10 int tls_connection_get_keys (void * *tls_ctx*, struct tls_connection * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

keys Structure of key/random data (filled on success)

Returns:

0 on success, -1 on failure

15.136.2.11 int tls_connection_get_read_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection read alert status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Number of times a fatal read (remote end reported error) has happened during this connection.

15.136.2.12 int tls_connection_get_write_alerts (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection write alert status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

Number of times a fatal write (locally detected error) has happened during this connection.

15.136.2.13 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

appl_data Pointer to application data pointer, or NULL if dropped

appl_data_len Pointer to variable that is set to appl_data length

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server

is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

15.136.2.14 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

15.136.2.15 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

key Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

key_len Length of session key material

Returns:

0 on success, -1 on failure

15.136.2.16 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

final 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

out_data Pointer to output buffer (encrypted TLS/IA data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

15.136.2.17 struct tls_connection* tls_connection_init (void * *tls_ctx*) [read]

Initialize a new TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Connection context data, conn for other function calls

15.136.2.18 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

15.136.2.19 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if current session used session resumption, 0 if not

15.136.2.20 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

Returns:

pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

15.136.2.21 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.136.2.22 int tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

tls_ia 1 = enable TLS/IA

Returns:

0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

15.136.2.23 int tls_connection_set_params (void * *tls_ctx*, struct tls_connection * *conn*, const struct tls_connection_params * *params*)

Set TLS connection parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

params Connection parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.136.2.24 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

verify_peer 1 = verify peer certificate

Returns:

0 on success, -1 on failure

15.136.2.25 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

15.136.2.26 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

Parameters:

tls_ctx TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

15.136.2.27 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

buf Buffer for the cipher name

buflen buf size

Returns:

0 on success, -1 on failure

Get the name of the currently used cipher.

15.136.2.28 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

15.136.2.29 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

params Global TLS parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.136.2.30 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

check_crl 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

Returns:

0 on success, -1 on failure

15.136.2.31 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

Parameters:

conf Configuration data for TLS library

Returns:

Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

15.137 src/crypto/tls_openssl.c File Reference

WPA Supplicant / SSL/TLS interface functions for openssl. `#include "includes.h"`

```
#include <openssl/ssl.h>
#include <openssl/err.h>
#include <openssl/pkcs12.h>
#include <openssl/x509v3.h>
#include <openssl/engine.h>
#include "common.h"
#include "tls.h"
```

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.
- struct tls_connection * tls_connection_init (void *ssl_ctx)
  Initialize a new TLS connection.
- void tls_connection_deinit (void *ssl_ctx, struct tls_connection *conn)
  Free TLS connection data.
- int tls_connection_established (void *ssl_ctx, struct tls_connection *conn)
  Has the TLS connection been completed?
- int tls_connection_shutdown (void *ssl_ctx, struct tls_connection *conn)
  Shutdown TLS connection.
- int tls_global_set_verify (void *ssl_ctx, int check_crl)
  Set global certificate verification options.
- int tls_connection_set_verify (void *ssl_ctx, struct tls_connection *conn, int verify_peer)

Set certificate verification options.

- int tls_connection_get_keys (void *ssl_ctx, struct tls_connection *conn, struct tls_keys *keys)

  Get master key and random data from TLS connection.

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  Use TLS-PRF to derive keying material.

- u8 * tls_connection_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  Process TLS handshake (client side).

- u8 * tls_connection_server_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  Process TLS handshake (server side).

- int tls_connection_encrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Encrypt data into TLS tunnel.

- int tls_connection_decrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Decrypt data from TLS tunnel.

- int tls_connection_resumed (void *ssl_ctx, struct tls_connection *conn)

  Was session resumption used.

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  Configure acceptable cipher suites.

- int tls_get_cipher (void *ssl_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  Get current cipher name.

- int tls_connection_enable_workaround (void *ssl_ctx, struct tls_connection *conn)

  Enable TLS workaround options.

- int tls_connection_get_failed (void *ssl_ctx, struct tls_connection *conn)

  Get connection failure status.

- int tls_connection_get_read_alerts (void *ssl_ctx, struct tls_connection *conn)

  Get connection read alert status.

- int tls_connection_get_write_alerts (void *ssl_ctx, struct tls_connection *conn)

  Get connection write alert status.

- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  Set TLS connection parameters.

- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)
  Set TLS parameters for all TLS connection.
- int tls_connection_get_keyblock_size (void *tls_ctx, struct tls_connection *conn)
  Get TLS key_block size.
- unsigned int tls_capabilities (void *tls_ctx)
  Get supported TLS capabilities.
- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)
  Set TLS/IA parameters.
- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)
  Send a TLS/IA PhaseFinished message.
- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)
  Has final phase been completed.
- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)
  Permute TLS/IA inner secret.
- int **tls_connection_set_session_ticket_cb** (void *tls_ctx, struct tls_connection *conn, tls_session_ticket_cb cb, void *ctx)

15.137.1 Detailed Description

WPA Supplicant / SSL/TLS interface functions for openssl.

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.137.2 Function Documentation

15.137.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

15.137.2.2 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.

15.137.2.3 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.

15.137.2.4 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.

15.137.2.5 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.

15.137.2.6 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.

15.137.2.7 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.

15.137.2.8 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

15.137.2.9 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

15.137.2.10 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.

15.137.2.11 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.

15.137.2.12 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

appl_data Pointer to application data pointer, or NULL if dropped

appl_data_len Pointer to variable that is set to appl_data length

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

15.137.2.13 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

15.137.2.14 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

key Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

key_len Length of session key material

Returns:

0 on success, -1 on failure

15.137.2.15 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

final 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

out_data Pointer to output buffer (encrypted TLS/IA data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

15.137.2.16 struct tls_connection* tls_connection_init (void * *tls_ctx*) [read]

Initialize a new TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Connection context data, conn for other function calls

15.137.2.17 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

15.137.2.18 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if current session used session resumption, 0 if not

15.137.2.19 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

Returns:

pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

15.137.2.20 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.137.2.21 int tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

tls_ia 1 = enable TLS/IA

Returns:

0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

15.137.2.22 int tls_connection_set_params (void * *tls_ctx*, struct tls_connection * *conn*, const struct tls_connection_params * *params*)

Set TLS connection parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

params Connection parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.137.2.23 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

verify_peer 1 = verify peer certificate

Returns:

0 on success, -1 on failure

15.137.2.24 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

15.137.2.25 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

Parameters:

tls_ctx TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

15.137.2.26 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

buf Buffer for the cipher name

buflen buf size

Returns:

0 on success, -1 on failure

Get the name of the currently used cipher.

15.137.2.27 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

15.137.2.28 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

params Global TLS parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.137.2.29 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

check_crl 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

Returns:

0 on success, -1 on failure

15.137.2.30 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

Parameters:

conf Configuration data for TLS library

Returns:

Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

15.138 src/crypto/tls_schannel.c File Reference

WPA Supplicant / SSL/TLS interface functions for Microsoft Schannel. `#include "includes.h"`

```
#include <windows.h>
#include <wincrypt.h>
#include <schannel.h>
#include <security.h>
#include <sspi.h>
#include "common.h"
#include "tls.h"
```

Data Structures

- struct tls_global
- struct tls_connection
- struct _SecPkgContext_EapKeyBlock

Defines

- #define **SECURITY_WIN32**
- #define **SECPKG_ATTR_EAP_KEY_BLOCK** 0x5b

Typedefs

- typedef struct _SecPkgContext_EapKeyBlock **SecPkgContext_EapKeyBlock**
- typedef struct _SecPkgContext_EapKeyBlock * **PSecPkgContext_EapKeyBlock**

Functions

- void * tls_init (const struct tls_config *conf)

  Initialize TLS library.

- void tls_deinit (void *ssl_ctx)

  Deinitialize TLS library.

- int tls_get_errors (void *ssl_ctx)

  Process pending errors.

- struct tls_connection * tls_connection_init (void *ssl_ctx)

  Initialize a new TLS connection.

- void tls_connection_deinit (void *ssl_ctx, struct tls_connection *conn)

  Free TLS connection data.

- int tls_connection_established (void *ssl_ctx, struct tls_connection *conn)

Has the TLS connection been completed?

- int tls_connection_shutdown (void *ssl_ctx, struct tls_connection *conn)

  Shutdown TLS connection.

- int tls_global_set_params (void *tls_ctx, const struct tls_connection_params *params)

  Set TLS parameters for all TLS connection.

- int tls_global_set_verify (void *ssl_ctx, int check_crl)

  Set global certificate verification options.

- int tls_connection_set_verify (void *ssl_ctx, struct tls_connection *conn, int verify_peer)

  Set certificate verification options.

- int tls_connection_get_keys (void *ssl_ctx, struct tls_connection *conn, struct tls_keys *keys)

  Get master key and random data from TLS connection.

- int tls_connection_prf (void *tls_ctx, struct tls_connection *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  Use TLS-PRF to derive keying material.

- u8 * tls_connection_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  Process TLS handshake (client side).

- u8 * tls_connection_server_handshake (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  Process TLS handshake (server side).

- int tls_connection_encrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Encrypt data into TLS tunnel.

- int tls_connection_decrypt (void *ssl_ctx, struct tls_connection *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Decrypt data from TLS tunnel.

- int tls_connection_resumed (void *ssl_ctx, struct tls_connection *conn)

  Was session resumption used.

- int tls_connection_set_cipher_list (void *tls_ctx, struct tls_connection *conn, u8 *ciphers)

  Configure acceptable cipher suites.

- int tls_get_cipher (void *ssl_ctx, struct tls_connection *conn, char *buf, size_t buflen)

  Get current cipher name.

- int tls_connection_enable_workaround (void *ssl_ctx, struct tls_connection *conn)

  Enable TLS workaround options.

- int tls_connection_client_hello_ext (void *ssl_ctx, struct tls_connection *conn, int ext_type, const u8 *data, size_t data_len)

Set TLS extension for ClientHello.

- int tls_connection_get_failed (void *ssl_ctx, struct tls_connection *conn)

  Get connection failure status.

- int tls_connection_get_read_alerts (void *ssl_ctx, struct tls_connection *conn)

  Get connection read alert status.

- int tls_connection_get_write_alerts (void *ssl_ctx, struct tls_connection *conn)

  Get connection write alert status.

- int tls_connection_set_params (void *tls_ctx, struct tls_connection *conn, const struct tls_connection_params *params)

  Set TLS connection parameters.

- unsigned int tls_capabilities (void *tls_ctx)

  Get supported TLS capabilities.

- int tls_connection_set_ia (void *tls_ctx, struct tls_connection *conn, int tls_ia)

  Set TLS/IA parameters.

- int tls_connection_ia_send_phase_finished (void *tls_ctx, struct tls_connection *conn, int final, u8 *out_data, size_t out_len)

  Send a TLS/IA PhaseFinished message.

- int tls_connection_ia_final_phase_finished (void *tls_ctx, struct tls_connection *conn)

  Has final phase been completed.

- int tls_connection_ia_permute_inner_secret (void *tls_ctx, struct tls_connection *conn, const u8 *key, size_t key_len)

  Permute TLS/IA inner secret.

15.138.1 Detailed Description

WPA Supplicant / SSL/TLS interface functions for Microsoft Schannel.

Copyright

15.138.2 Function Documentation

15.138.2.1 unsigned int tls_capabilities (void * *tls_ctx*)

Get supported TLS capabilities.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Bit field of supported TLS capabilities (TLS_CAPABILITY_*)

15.138.2.2 int tls_connection_client_hello_ext (void * *tls_ctx*, struct tls_connection * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ext_type Extension type

data Extension payload (NULL to remove extension)

data_len Extension payload length

Returns:

0 on success, -1 on failure

15.138.2.3 int tls_connection_decrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Pointer to input buffer (encrypted TLS data)

in_len Input buffer length

out_data Pointer to output buffer (decrypted data from TLS tunnel)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

15.138.2.4 void tls_connection_deinit (void * *tls_ctx*, struct tls_connection * *conn*)

Free TLS connection data.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Release all resources allocated for TLS connection.

15.138.2.5 int tls_connection_enable_workaround (void * *tls_ctx*, struct tls_connection * *conn*)

Enable TLS workaround options.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

This function is used to enable connection-specific workaround options for buffer SSL/TLS implementations.

15.138.2.6 int tls_connection_encrypt (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Pointer to plaintext data to be encrypted

in_len Input buffer length

out_data Pointer to output buffer (encrypted TLS data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

15.138.2.7 int tls_connection_established (void * *tls_ctx*, struct tls_connection * *conn*)

Has the TLS connection been completed?

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if TLS connection has been completed, 0 if not.

15.138.2.8 int tls_connection_get_failed (void * *tls_ctx*, struct tls_connection * *conn*)

Get connection failure status.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns >0 if connection has failed, 0 if not.

15.138.2.9 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

15.138.2.10 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.

15.138.2.11 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.

15.138.2.12 u8* tls_connection_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake (client side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

appl_data Pointer to application data pointer, or NULL if dropped

appl_data_len Pointer to variable that is set to appl_data length

Returns:

Pointer to output data, NULL on failure

Caller is responsible for freeing returned output data. If the final handshake message includes application data, this is decrypted and appl_data (if not NULL) is set to point this data. Caller is responsible for freeing appl_data.

This function is used during TLS handshake. The first call is done with in_data == NULL and the library is expected to return ClientHello packet. This packet is then send to the server and a response from server is given to TLS library by calling this function again with in_data pointing to the TLS message from the server.

If the TLS handshake fails, this function may return NULL. However, if the TLS library has a TLS alert to send out, that should be returned as the output data. In this case, tls_connection_get_failed() must return failure (> 0).

tls_connection_established() should return 1 once the TLS handshake has been completed successfully.

15.138.2.13 int tls_connection_ia_final_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*)

Has final phase been completed.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if valid FinalPhaseFinished has been received, 0 if not, or -1 on failure

15.138.2.14 int tls_connection_ia_permute_inner_secret (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *key*, size_t *key_len*)

Permute TLS/IA inner secret.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

key Session key material (session_key vectors with 2-octet length), or NULL if no session key was generating in the current phase

key_len Length of session key material

Returns:

0 on success, -1 on failure

15.138.2.15 int tls_connection_ia_send_phase_finished (void * *tls_ctx*, struct tls_connection * *conn*, int *final*, u8 * *out_data*, size_t *out_len*)

Send a TLS/IA PhaseFinished message.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

final 1 = FinalPhaseFinished, 0 = IntermediatePhaseFinished

out_data Pointer to output buffer (encrypted TLS/IA data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data on success, -1 on failure

This function is used to send the TLS/IA end phase message, e.g., when the EAP server completes EAP-TTLSv1.

15.138.2.16 struct tls_connection* tls_connection_init (void * *tls_ctx*) [read]

Initialize a new TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Connection context data, conn for other function calls

15.138.2.17 int tls_connection_prf (void * *tls_ctx*, struct tls_connection * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

This function is optional to implement if tls_connection_get_keys() provides access to master secret and server/client random values. If these values are not exported from the TLS library, tls_connection_prf() is required so that further keying material can be derived from the master secret. If not implemented, the function will still need to be defined, but it can just return -1. Example implementation of this function is in tls_prf() function when it is called with seed set to client_random|server_random (or server_random|client_random).

15.138.2.18 int tls_connection_resumed (void * *tls_ctx*, struct tls_connection * *conn*)

Was session resumption used.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

1 if current session used session resumption, 0 if not

15.138.2.19 u8* tls_connection_server_handshake (void * *tls_ctx*, struct tls_connection * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake (server side).

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

Returns:

pointer to output data, NULL on failure

Caller is responsible for freeing returned output data.

15.138.2.20 int tls_connection_set_cipher_list (void * *tls_ctx*, struct tls_connection * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.138.2.21 int tls_connection_set_ia (void * *tls_ctx*, struct tls_connection * *conn*, int *tls_ia*)

Set TLS/IA parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

tls_ia 1 = enable TLS/IA

Returns:

0 on success, -1 on failure

This function is used to configure TLS/IA in server mode where tls_connection_set_params() is not used.

15.138.2.22 int tls_connection_set_params (void * *tls_ctx*, struct tls_connection * *conn*, const struct tls_connection_params * *params*)

Set TLS connection parameters.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

params Connection parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.138.2.23 int tls_connection_set_verify (void * *tls_ctx*, struct tls_connection * *conn*, int *verify_peer*)

Set certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

verify_peer 1 = verify peer certificate

Returns:

0 on success, -1 on failure

15.138.2.24 int tls_connection_shutdown (void * *tls_ctx*, struct tls_connection * *conn*)

Shutdown TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

Returns:

0 on success, -1 on failure

Shutdown current TLS connection without releasing all resources. New connection can be started by using the same conn without having to call tls_connection_init() or setting certificates etc. again. The new connection should try to use session resumption.

15.138.2.25 void tls_deinit (void * *tls_ctx*)

Deinitialize TLS library.

Parameters:

tls_ctx TLS context data from tls_init()

Called once during program shutdown and once for each RSN pre-authentication session. If global library deinitialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global deinitialization only when moving from 1 to 0 references.

15.138.2.26 int tls_get_cipher (void * *tls_ctx*, struct tls_connection * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

Parameters:

tls_ctx TLS context data from tls_init()

conn Connection context data from tls_connection_init()

buf Buffer for the cipher name

buflen buf size

Returns:

0 on success, -1 on failure

Get the name of the currently used cipher.

15.138.2.27 int tls_get_errors (void * *tls_ctx*)

Process pending errors.

Parameters:

tls_ctx TLS context data from tls_init()

Returns:

Number of found error, 0 if no errors detected.

Process all pending TLS errors.

15.138.2.28 int tls_global_set_params (void * *tls_ctx*, const struct tls_connection_params * *params*)

Set TLS parameters for all TLS connection.

Parameters:

tls_ctx TLS context data from tls_init()

params Global TLS parameters

Returns:

0 on success, -1 on failure, TLS_SET_PARAMS_ENGINE_PRV_INIT_FAILED (-2) on possible PIN error causing PKCS#11 engine failure, or TLS_SET_PARAMS_ENGINE_PRV_VERIFY_FAILED (-3) on failure to verify the PKCS#11 engine private key.

15.138.2.29 int tls_global_set_verify (void * *tls_ctx*, int *check_crl*)

Set global certificate verification options.

Parameters:

tls_ctx TLS context data from tls_init()

check_crl 0 = do not verify CRLs, 1 = verify CRL for the user certificate, 2 = verify CRL for all certificates

Returns:

0 on success, -1 on failure

15.138.2.30 void* tls_init (const struct tls_config * *conf*)

Initialize TLS library.

Parameters:

conf Configuration data for TLS library

Returns:

Context data to be used as tls_ctx in calls to other functions, or NULL on failure.

Called once during program startup and once for each RSN pre-authentication session. In other words, there can be two concurrent TLS contexts. If global library initialization is needed (i.e., one that is shared between both authentication types), the TLS library wrapper should maintain a reference counter and do global initialization only when moving from 0 to 1 reference.

15.139 src/drivers/driver.h File Reference

WPA Supplicant - driver interface definition. `#include "defs.h"`

Data Structures

- struct hostapd_channel_data
- struct hostapd_rate_data
- struct hostapd_hw_modes
- struct wpa_scan_res

  Scan result for an BSS/IBSS.

- struct wpa_scan_results

  Scan results.

- struct wpa_interface_info

  Network interface information.

- struct wpa_driver_scan_params

  Scan parameters.

- struct wpa_driver_scan_params::wpa_driver_scan_ssid

  SSIDs to scan for.

- struct wpa_driver_auth_params

  Authentication parameters.

- struct wpa_driver_associate_params

  Association parameters.

- struct wpa_driver_capa

  Driver capability information.

- struct ieee80211_rx_status
- struct hostap_sta_driver_data
- struct hostapd_sta_add_params
- struct hostapd_freq_params
- struct wpa_init_params
- struct wpa_driver_ops

  Driver interface API definition.

- union wpa_event_data
- struct wpa_event_data::assoc_info

  Data for EVENT_ASSOC and EVENT_ASSOCINFO events.

- struct wpa_event_data::michael_mic_failure

  Data for EVENT_MICHAEL_MIC_FAILURE.

- struct wpa_event_data::interface_status

Data for EVENT_INTERFACE_STATUS.

- struct wpa_event_data::pmkid_candidate
  Data for EVENT_PMKID_CANDIDATE.
- struct wpa_event_data::stkstart
  Data for EVENT_STKSTART.
- struct wpa_event_data::ft_ies
  FT information elements (EVENT_FT_RESPONSE).
- struct wpa_event_data::ibss_rsn_start
  Data for EVENT_IBSS_RSN_START.
- struct wpa_event_data::auth_info
  Data for EVENT_AUTH events.
- struct wpa_event_data::assoc_reject
  Data for EVENT_ASSOC_REJECT events.
- struct wpa_event_data::timeout_event
- struct hostapd_frame_info

Defines

- #define **WPA_SUPPLICANT_DRIVER_VERSION** 4
- #define **HOSTAPD_CHAN_DISABLED** 0x00000001
- #define **HOSTAPD_CHAN_PASSIVE_SCAN** 0x00000002
- #define **HOSTAPD_CHAN_NO_IBSS** 0x00000004
- #define **HOSTAPD_CHAN_RADAR** 0x00000008
- #define **HOSTAPD_RATE_ERP** 0x00000001
- #define **HOSTAPD_RATE_BASIC** 0x00000002
- #define **HOSTAPD_RATE_PREAMBLE2** 0x00000004
- #define **HOSTAPD_RATE_SUPPORTED** 0x00000010
- #define **HOSTAPD_RATE_OFDM** 0x00000020
- #define **HOSTAPD_RATE_CCK** 0x00000040
- #define **HOSTAPD_RATE_MANDATORY** 0x00000100
- #define **AUTH_ALG_OPEN_SYSTEM** 0x01
- #define **AUTH_ALG_SHARED_KEY** 0x02
- #define **AUTH_ALG_LEAP** 0x04
- #define **AUTH_ALG_FT** 0x08
- #define **IEEE80211_MODE_INFRA** 0
- #define **IEEE80211_MODE_IBSS** 1
- #define **IEEE80211_MODE_AP** 2
- #define **IEEE80211_CAP_ESS** 0x0001
- #define **IEEE80211_CAP_IBSS** 0x0002
- #define **IEEE80211_CAP_PRIVACY** 0x0010
- #define **WPA_SCAN_QUAL_INVALID** BIT(0)
- #define **WPA_SCAN_NOISE_INVALID** BIT(1)
- #define **WPA_SCAN_LEVEL_INVALID** BIT(2)

- #define **WPA_SCAN_LEVEL_DBM** BIT(3)
- #define **WPAS_MAX_SCAN_SSIDS** 4
- #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_KEY_MGMT_FT** 0x00000020
- #define **WPA_DRIVER_CAPA_KEY_MGMT_FT_PSK** 0x00000040
- #define **WPA_DRIVER_CAPA_ENC_WEP40** 0x00000001
- #define **WPA_DRIVER_CAPA_ENC_WEP104** 0x00000002
- #define **WPA_DRIVER_CAPA_ENC_TKIP** 0x00000004
- #define **WPA_DRIVER_CAPA_ENC_CCMP** 0x00000008
- #define **WPA_DRIVER_AUTH_OPEN** 0x00000001
- #define **WPA_DRIVER_AUTH_SHARED** 0x00000002
- #define **WPA_DRIVER_AUTH_LEAP** 0x00000004
- #define **WPA_DRIVER_FLAGS_DRIVER_IE** 0x00000001
- #define **WPA_DRIVER_FLAGS_SET_KEYS_AFTER_ASSOC** 0x00000002
- #define **WPA_DRIVER_FLAGS_USER_SPACE_MLME** 0x00000004
- #define **WPA_DRIVER_FLAGS_4WAY_HANDSHAKE** 0x00000008
- #define **WPA_DRIVER_FLAGS_WIRED** 0x00000010
- #define **WPA_DRIVER_FLAGS_SME** 0x00000020
- #define **WPA_DRIVER_FLAGS_AP** 0x00000040
- #define **WPA_DRIVER_FLAGS_SET_KEYS_AFTER_ASSOC_DONE** 0x00000080
- #define **WPA_IE_VENDOR_TYPE** 0x0050f201
- #define **WPS_IE_VENDOR_TYPE** 0x0050f204

Enumerations

- enum **hostapd_driver_if_type** { **HOSTAPD_IF_VLAN** }
- enum wpa_event_type {

  EVENT_ASSOC, EVENT_DISASSOC, EVENT_MICHAEL_MIC_FAILURE, EVENT_SCAN_-RESULTS,

  EVENT_ASSOCINFO, EVENT_INTERFACE_STATUS, EVENT_PMKID_CANDIDATE, EVENT_STKSTART,

  EVENT_FT_RESPONSE, EVENT_IBSS_RSN_START, EVENT_AUTH, EVENT_DEAUTH,

  EVENT_ASSOC_REJECT, EVENT_AUTH_TIMED_OUT, EVENT_ASSOC_TIMED_OUT }

Functions

- void wpa_supplicant_event (void *ctx, wpa_event_type event, union wpa_event_data *data)

  Report a driver event for wpa_supplicant.

- void wpa_supplicant_rx_eapol (void *ctx, const u8 *src_addr, const u8 *buf, size_t len)

  Deliver a received EAPOL frame to wpa_supplicant.

- void **wpa_supplicant_sta_rx** (void *ctx, const u8 *buf, size_t len, struct ieee80211_rx_status *rx_-status)

- const u8 * **wpa_scan_get_ie** (const struct wpa_scan_res *res, u8 ie)
- const u8 * **wpa_scan_get_vendor_ie** (const struct wpa_scan_res *res, u32 vendor_type)
- struct wpabuf * **wpa_scan_get_vendor_ie_multi** (const struct wpa_scan_res *res, u32 vendor_type)

- int **wpa_scan_get_max_rate** (const struct wpa_scan_res *res)
- void **wpa_scan_results_free** (struct wpa_scan_results *res)
- void **wpa_scan_sort_results** (struct wpa_scan_results *res)
- void hostapd_new_assoc_sta (struct hostapd_data *hapd, struct sta_info *sta, int reassoc)

  Notify that a new station associated with the AP.

- void **hostapd_tx_status** (struct hostapd_data *hapd, const u8 *addr, const u8 *buf, size_t len, int ack)
- void **hostapd_rx_from_unknown_sta** (struct hostapd_data *hapd, const struct ieee80211_hdr *hdr, size_t len)
- int **hostapd_notif_assoc** (struct hostapd_data *hapd, const u8 *addr, const u8 *ie, size_t ielen)
- void **hostapd_notif_disassoc** (struct hostapd_data *hapd, const u8 *addr)
- void **hostapd_eapol_receive** (struct hostapd_data *hapd, const u8 *sa, const u8 *buf, size_t len)
- void **hostapd_mgmt_rx** (struct hostapd_data *hapd, u8 *buf, size_t len, u16 stype, struct hostapd_frame_info *fi)
- void **hostapd_mgmt_tx_cb** (struct hostapd_data *hapd, u8 *buf, size_t len, u16 stype, int ok)
- void **hostapd_michael_mic_failure** (struct hostapd_data *hapd, const u8 *addr)
- struct hostapd_data * **hostapd_sta_get_bss** (struct hostapd_data *hapd, const u8 *addr)
- void **hostapd_probe_req_rx** (struct hostapd_data *hapd, const u8 *sa, const u8 *ie, size_t ie_len)

15.139.1 Detailed Description

WPA Supplicant - driver interface definition.

Copyright

Copyright (c) 2003-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.139.2 Enumeration Type Documentation

15.139.2.1 enum wpa_event_type

enum wpa_event_type - Event type for wpa_supplicant_event() calls

Enumerator:

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. If the driver supports separate deauthentication event, EVENT_DISASSOC should only be used for disassociation and EVENT_DEAUTH for deauthentication.

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 encyption key and to initiate TKIP countermeasures if needed.

EVENT_SCAN_RESULTS Scan results available. This event must be called whenever scan results are available to be fetched with struct wpa_driver_ops::get_scan_results(). This event is expected to be used some time after struct wpa_driver_ops::scan() is called. If the driver provides an unsolicited event when the scan has been completed, this event can be used to trigger EVENT_SCAN_RESULTS call. If such event is not available from the driver, the driver wrapper code is expected to use a registered timeout to generate EVENT_SCAN_RESULTS call after the time that the scan is expected to be completed.

EVENT_ASSOCINFO Report optional extra information for association. This event can be used to report extra association information for EVENT_ASSOC processing. This extra information includes IEs from association frames and Beacon/Probe Response frames in union wpa_event_data::assoc_info. EVENT_ASSOCINFO must be send just before EVENT_ASSOC. Alternatively, the driver interface can include assoc_info data in the EVENT_ASSOC call if it has all the information available at the same point.

EVENT_INTERFACE_STATUS Report interface status changes. This optional event can be used to report changes in interface status (interface added/removed) using union wpa_event_data::interface_status. This can be used to trigger wpa_supplicant to stop and re-start processing for the interface, e.g., when a cardbus card is ejected/inserted.

EVENT_PMKID_CANDIDATE Report a candidate AP for pre-authentication. This event can be used to inform wpa_supplicant about candidates for RSN (WPA2) pre-authentication. If wpa_supplicant is not responsible for scan request (ap_scan=2 mode), this event is required for pre-authentication. If wpa_supplicant is performing scan request (ap_scan=1), this event is optional since scan results can be used to add pre-authentication candidates. union wpa_event_data::pmkid_candidate is used to report the BSSID of the candidate and priority of the candidate, e.g., based on the signal strength, in order to try to pre-authenticate first with candidates that are most likely targets for re-association.

EVENT_PMKID_CANDIDATE can be called whenever the driver has updates on the candidate list. In addition, it can be called for the current AP and APs that have existing PMKSA cache entries. wpa_supplicant will automatically skip pre-authentication in cases where a valid PMKSA exists. When more than one candidate exists, this event should be generated once for each candidate.

Driver will be notified about successful pre-authentication with struct wpa_driver_ops::add_pmkid() calls.

EVENT_STKSTART Request STK handshake (MLME-STKSTART.request). This event can be used to inform wpa_supplicant about desire to set up secure direct link connection between two stations as defined in IEEE 802.11e with a new PeerKey mechanism that replaced the original STAKey negotiation. The caller will need to set peer address for the event.

EVENT_FT_RESPONSE Report FT (IEEE 802.11r) response IEs. The driver is expected to report the received FT IEs from FT authentication sequence from the AP. The FT IEs are included in the extra information in union wpa_event_data::ft_ies.

EVENT_IBSS_RSN_START Request RSN authentication in IBSS. The driver can use this event to inform wpa_supplicant about a STA in an IBSS with which protected frames could be exchanged. This event starts RSN authentication with the other STA to authenticate the STA and set up encryption keys with it.

EVENT_AUTH Authentication result. This event should be called when authentication attempt has been completed. This is only used if the driver supports separate authentication step (struct wpa_-driver_ops::authenticate). Information about authentication result is included in union wpa_-event_data::auth.

EVENT_DEAUTH Authentication lost. This event should be called when authentication is lost either due to receiving deauthenticate frame from the AP or when sending that frame to the current AP.

EVENT_ASSOC_REJECT Association rejected. This event should be called when (re)association attempt has been rejected by the AP. Information about authentication result is included in union wpa_event_data::assoc_reject.

EVENT_AUTH_TIMED_OUT Authentication timed out.

EVENT_ASSOC_TIMED_OUT Association timed out.

15.139.3 Function Documentation

15.139.3.1 void hostapd_new_assoc_sta (struct hostapd_data * *hapd*, struct sta_info * *sta*, int *reassoc*)

Notify that a new station associated with the AP.

Parameters:

hapd Pointer to BSS data

sta Pointer to the associated STA data

reassoc 1 to indicate this was a re-association; 0 = first association

This function will be called whenever a station associates with the AP. It can be called from ieee802_11.c for drivers that export MLME to hostapd and from driver_*.c for drivers that take care of management frames (IEEE 802.11 authentication and association) internally.

15.139.3.2 void wpa_supplicant_event (void * *ctx*, wpa_event_type *event*, union wpa_event_data * *data*)

Report a driver event for wpa_supplicant.

Parameters:

ctx Context pointer (wpa_s); 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.

15.139.3.3 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); this is the ctx variable registered with struct wpa_driver_ops::init()

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. Most driver interfaces rely on more generic OS mechanism for receiving frames through l2_packet, but if such a mechanism is not available, the driver wrapper may take care of received EAPOL frames and deliver them to the core supplicant code by calling this function.

15.140 src/drivers/driver_atheros.c File Reference

hostapd / Driver interaction with Atheros driver `#include "includes.h"`

```
#include <net/if.h>
#include <sys/ioctl.h>
#include "common.h"
#include <net80211/ieee80211.h>
#include <net80211/_ieee80211.h>
#include <net80211/ieee80211_crypto.h>
#include <net80211/ieee80211_ioctl.h>
#include "wireless_copy.h"
#include "../hostapd/hostapd.h"
#include "../hostapd/config.h"
#include "../hostapd/sta_flags.h"
#include "driver.h"
#include "eloop.h"
#include "priv_netlink.h"
#include "l2_packet/l2_packet.h"
#include "../hostapd/wps_hostapd.h"
#include "ieee802_11_defs.h"
```

Data Structures

- struct madwifi_driver_data

Defines

- #define **_BYTE_ORDER** _LITTLE_ENDIAN
- #define **ATH_WPS_IE**
- #define **madwifi_set_wps_beacon_ie** NULL
- #define **madwifi_set_wps_probe_resp_ie** NULL

Variables

- struct wpa_driver_ops **wpa_driver_atheros_ops**

15.140.1 Detailed Description

hostapd / Driver interaction with Atheros driver

Copyright

Copyright (c) 2004, Sam Leffler <sam@errno.com> Copyright (c) 2004, Video54 Technologies Copyright (c) 2005-2007, Jouni Malinen <j@w1.fi> Copyright (c) 2009, Atheros Communications

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.140.2 Variable Documentation

15.140.2.1 struct wpa_driver_ops wpa_driver_atheros_ops

Initial value:

```
{
        .name                   = "atheros",
        .hapd_init              = madwifi_init,
        .deinit                 = madwifi_deinit,
        .set_ieee8021x          = madwifi_set_ieee8021x,
        .set_privacy            = madwifi_set_privacy,
        .set_key                = madwifi_set_key,
        .get_seqnum             = madwifi_get_seqnum,
        .flush                  = madwifi_flush,
        .set_generic_elem       = madwifi_set_opt_ie,
        .sta_set_flags          = madwifi_sta_set_flags,
        .read_sta_data          = madwifi_read_sta_driver_data,
        .hapd_send_eapol        = madwifi_send_eapol,
        .sta_disassoc           = madwifi_sta_disassoc,
        .sta_deauth             = madwifi_sta_deauth,
        .hapd_set_ssid          = madwifi_set_ssid,
        .hapd_get_ssid          = madwifi_get_ssid,
        .set_countermeasures    = madwifi_set_countermeasures,
        .sta_clear_stats        = madwifi_sta_clear_stats,
        .commit                 = madwifi_commit,
        .set_wps_beacon_ie      = madwifi_set_wps_beacon_ie,
        .set_wps_probe_resp_ie  = madwifi_set_wps_probe_resp_ie,
}
```

15.141 src/drivers/driver_atmel.c File Reference

WPA Supplicant - Driver interaction with Atmel Wireless LAN drivers. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
```

Data Structures

- struct wpa_driver_atmel_data
- struct atmel_param

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

- struct wpa_driver_ops **wpa_driver_atmel_ops**

15.141.1 Detailed Description

WPA Supplicant - Driver interaction with Atmel Wireless LAN drivers.

Copyright

Copyright (c) 2000-2005, ATMEL Corporation Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.141.2 Variable Documentation

15.141.2.1 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_key = wpa_driver_atmel_set_key,
        .init = wpa_driver_atmel_init,
        .deinit = wpa_driver_atmel_deinit,
        .set_countermeasures = wpa_driver_atmel_set_countermeasures,
        .scan2 = wpa_driver_atmel_scan,
        .get_scan_results2 = wpa_driver_atmel_get_scan_results,
        .deauthenticate = wpa_driver_atmel_deauthenticate,
        .disassociate = wpa_driver_atmel_disassociate,
        .associate = wpa_driver_atmel_associate,
        .set_operstate = wpa_driver_atmel_set_operstate,
}
```

15.142 src/drivers/driver_broadcom.c File Reference

WPA Supplicant - driver interaction with old Broadcom wl.o driver. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "common.h"
#include <linux/if_packet.h>
#include <linux/if_ether.h>
#include <net/if.h>
#include <typedefs.h>
#include <wlioctl.h>
#include "driver.h"
#include "eloop.h"
```

Data Structures

- struct wpa_driver_broadcom_data
- struct wlc_deauth_t
- struct bss_ie_hdr

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

- struct bss_ie_hdr **packed**
- struct wpa_driver_ops **wpa_driver_broadcom_ops**

15.142.1 Detailed Description

WPA Supplicant - driver interaction with old Broadcom wl.o driver.

Copyright

Copyright (c) 2004, Nikki Chumkov <nikki@gattaca.ru> Copyright (c) 2004, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Please note that the newer Broadcom driver ("hybrid Linux driver") supports Linux wireless extensions and does not need (or even work) with this old driver wrapper. Use driver_wext.c with that driver.

15.142.2 Variable Documentation

15.142.2.1 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_key = wpa_driver_broadcom_set_key,
        .init = wpa_driver_broadcom_init,
        .deinit = wpa_driver_broadcom_deinit,
        .set_countermeasures = wpa_driver_broadcom_set_countermeasures,
        .scan2 = wpa_driver_broadcom_scan,
        .get_scan_results2 = wpa_driver_broadcom_get_scan_results,
        .deauthenticate = wpa_driver_broadcom_deauthenticate,
        .disassociate = wpa_driver_broadcom_disassociate,
        .associate = wpa_driver_broadcom_associate,
}
```

15.143 src/drivers/driver_bsd.c File Reference

WPA Supplicant - driver interaction with BSD net80211 layer. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "common.h"
#include "driver.h"
#include "eloop.h"
#include "ieee802_11_defs.h"
#include <net/if.h>
#include <net/ethernet.h>
#include <net/route.h>
#include <net80211/ieee80211.h>
#include <net80211/ieee80211_crypto.h>
#include <net80211/ieee80211_ioctl.h>
```

Data Structures

- struct wpa_driver_bsd_data

Defines

- #define **GETPARAM**(drv, param, v) (((v) = get80211param(drv, param)) != -1)

Functions

- struct wpa_scan_results * **wpa_driver_bsd_get_scan_results2** (void *priv)

Variables

- struct wpa_driver_ops **wpa_driver_bsd_ops**

15.143.1 Detailed Description

WPA Supplicant - driver interaction with BSD net80211 layer.

Copyright

15.143.2 Variable Documentation

15.143.2.1 struct wpa_driver_ops wpa_driver_bsd_ops

Initial value:

```
{
        .name                   = "bsd",
        .desc                   = "BSD 802.11 support",
        .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_key                = wpa_driver_bsd_set_key,
        .set_countermeasures    = wpa_driver_bsd_set_countermeasures,
        .scan2                  = wpa_driver_bsd_scan,
        .get_scan_results2      = wpa_driver_bsd_get_scan_results2,
        .deauthenticate         = wpa_driver_bsd_deauthenticate,
        .disassociate           = wpa_driver_bsd_disassociate,
        .associate              = wpa_driver_bsd_associate,
}
```

15.144 src/drivers/driver_hostap.c File Reference

Driver interaction with Linux Host AP driver. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
#include "eloop.h"
#include "driver_hostap.h"
```

Data Structures

- struct wpa_driver_hostap_data

Variables

- struct wpa_driver_ops **wpa_driver_hostap_ops**

15.144.1 Detailed Description

Driver interaction with Linux Host AP driver.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.145 src/drivers/driver_hostap.h File Reference

Driver interaction with Linux Host AP driver.

Data Structures

- struct prism2_download_param
- struct prism2_download_param::prism2_download_area
- struct prism2_hostapd_param

Defines

- #define **PRISM2_IOCTL_PRISM2_PARAM** (SIOCIWFIRSTPRIV + 0)
- #define **PRISM2_IOCTL_GET_PRISM2_PARAM** (SIOCIWFIRSTPRIV + 1)
- #define **PRISM2_IOCTL_WRITEMIF** (SIOCIWFIRSTPRIV + 2)
- #define **PRISM2_IOCTL_READMIF** (SIOCIWFIRSTPRIV + 3)
- #define **PRISM2_IOCTL_MONITOR** (SIOCIWFIRSTPRIV + 4)
- #define **PRISM2_IOCTL_RESET** (SIOCIWFIRSTPRIV + 6)
- #define **PRISM2_IOCTL_INQUIRE** (SIOCIWFIRSTPRIV + 8)
- #define **PRISM2_IOCTL_WDS_ADD** (SIOCIWFIRSTPRIV + 10)
- #define **PRISM2_IOCTL_WDS_DEL** (SIOCIWFIRSTPRIV + 12)
- #define **PRISM2_IOCTL_SET_RID_WORD** (SIOCIWFIRSTPRIV + 14)
- #define **PRISM2_IOCTL_MACCMD** (SIOCIWFIRSTPRIV + 16)
- #define **PRISM2_IOCTL_ADDMAC** (SIOCIWFIRSTPRIV + 18)
- #define **PRISM2_IOCTL_DELMAC** (SIOCIWFIRSTPRIV + 20)
- #define **PRISM2_IOCTL_KICKMAC** (SIOCIWFIRSTPRIV + 22)
- #define **PRISM2_IOCTL_DOWNLOAD** (SIOCDEVPRIVATE + 13)
- #define **PRISM2_IOCTL_HOSTAPD** (SIOCDEVPRIVATE + 14)
- #define **PRISM2_MAX_DOWNLOAD_AREA_LEN** 131072
- #define **PRISM2_MAX_DOWNLOAD_LEN** 262144
- #define **PRISM2_HOSTAPD_MAX_BUF_SIZE** 1024
- #define **PRISM2_HOSTAPD_RID_HDR_LEN** ((size_t) (&((struct prism2_hostapd_param *) 0)->u.rid.data))
- #define **PRISM2_HOSTAPD_GENERIC_ELEMENT_HDR_LEN** ((size_t) (&((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** BIT(0)
- #define **HOSTAP_CRYPT_FLAG_PERMANENT** BIT(1)
- #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 { **HOSTAP_ANTSEL_DO_NOT_TOUCH** = 0, **HOSTAP_ANTSEL_DIVERSITY** = 1, **HOSTAP_ANTSEL_LOW** = 2, **HOSTAP_ANTSEL_HIGH** = 3 }
- enum {

  AP_MAC_CMD_POLICY_OPEN = 0, **AP_MAC_CMD_POLICY_ALLOW** = 1, **AP_MAC_-CMD_POLICY_DENY** = 2, **AP_MAC_CMD_FLUSH** = 3,

  AP_MAC_CMD_KICKALL = 4 }
- enum {

  PRISM2_DOWNLOAD_VOLATILE = 1, **PRISM2_DOWNLOAD_NON_VOLATILE** = 3, **PRISM2_DOWNLOAD_VOLATILE_GENESIS** = 4, **PRISM2_DOWNLOAD_VOLATILE_-PERSISTENT** = 5,

  PRISM2_DOWNLOAD_VOLATILE_GENESIS_PERSISTENT = 6 }
- 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 }

15.145.1 Detailed Description

Driver interaction with Linux Host AP driver.

Copyright

Copyright (c) 2002-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.146 src/drivers/driver_iphone.m File Reference

WPA Supplicant - iPhone/iPod touch Apple80211 driver interface. `#include "includes.h"`

```
#include <CoreFoundation/CoreFoundation.h>
#include "common.h"
#include "driver.h"
#include "eloop.h"
#include "ieee802_11_defs.h"
#include "MobileApple80211.h"
```

Data Structures

- struct wpa_driver_iphone_data

Defines

- #define **Boolean** __DummyBoolean

Variables

- struct wpa_driver_ops **wpa_driver_iphone_ops**

15.146.1 Detailed Description

WPA Supplicant - iPhone/iPod touch Apple80211 driver interface.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.146.2 Variable Documentation

15.146.2.1 struct wpa_driver_ops wpa_driver_iphone_ops

Initial value:

```
{
        .name = "iphone",
        .desc = "iPhone/iPod touch Apple80211 driver",
        .get_ssid = wpa_driver_iphone_get_ssid,
        .get_bssid = wpa_driver_iphone_get_bssid,
        .init = wpa_driver_iphone_init,
```

```
        .deinit = wpa_driver_iphone_deinit,
        .scan = wpa_driver_iphone_scan,
        .get_scan_results = wpa_driver_iphone_get_scan_results,
        .associate = wpa_driver_iphone_associate,
        .set_key = wpa_driver_iphone_set_key,
        .get_capa = wpa_driver_iphone_get_capa,
}
```

15.147 src/drivers/driver_ipw.c File Reference

WPA Supplicant - driver interaction with Linux ipw2100/2200 drivers. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
```

Data Structures

- struct wpa_driver_ipw_data
- struct ipw_param

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

- struct wpa_driver_ops **wpa_driver_ipw_ops**

15.147.1 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 <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

Please note that ipw2100/2200 drivers change to use generic Linux wireless extensions if the kernel includes support for WE-18 or newer (Linux 2.6.13 or newer). driver_wext.c should be used in those cases.

15.147.2 Variable Documentation

15.147.2.1 struct wpa_driver_ops wpa_driver_ipw_ops

Initial value:

```
{
        .name = "ipw",
        .desc = "Intel ipw2100/2200 driver (old; use wext with Linux 2.6.13 "
        "or newer)",
        .get_bssid = wpa_driver_ipw_get_bssid,
        .get_ssid = wpa_driver_ipw_get_ssid,
        .set_key = wpa_driver_ipw_set_key,
        .set_countermeasures = wpa_driver_ipw_set_countermeasures,
        .scan2 = wpa_driver_ipw_scan,
        .get_scan_results2 = wpa_driver_ipw_get_scan_results,
        .deauthenticate = wpa_driver_ipw_deauthenticate,
        .disassociate = wpa_driver_ipw_disassociate,
        .associate = wpa_driver_ipw_associate,
        .init = wpa_driver_ipw_init,
        .deinit = wpa_driver_ipw_deinit,
        .set_operstate = wpa_driver_ipw_set_operstate,
}
```

15.148 src/drivers/driver_madwifi.c File Reference

WPA Supplicant - driver interaction with MADWIFI 802.11 driver. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
#include "eloop.h"
#include "ieee802_11_defs.h"
#include "wireless_copy.h"
#include <include/compat.h>
#include <net80211/ieee80211.h>
#include <net80211/ieee80211_crypto.h>
#include <net80211/ieee80211_ioctl.h>
```

Data Structures

- struct wpa_driver_madwifi_data

Variables

- struct wpa_driver_ops **wpa_driver_madwifi_ops**

15.148.1 Detailed Description

WPA Supplicant - driver interaction with MADWIFI 802.11 driver.

Copyright

Copyright (c) 2004, Sam Leffler <sam@errno.com> Copyright (c) 2004, Video54 Technologies Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

While this driver wrapper supports both AP (hostapd) and station (wpa_supplicant) operations, the station side is deprecated and driver_wext.c should be used instead. This driver wrapper should only be used with hostapd for AP mode functionality.

15.149 src/drivers/driver_ndis.c File Reference

WPA Supplicant - Windows/NDIS driver interface. `#include "includes.h"`

```
#include <Packet32.h>
#include <ntddndis.h>
#include "common.h"
#include "driver.h"
#include "eloop.h"
#include "ieee802_11_defs.h"
#include "driver_ndis.h"
```

Data Structures

- struct NDIS_802_11_SSID
- struct NDIS_802_11_CONFIGURATION_FH
- struct NDIS_802_11_CONFIGURATION
- struct NDIS_WLAN_BSSID_EX
- struct NDIS_802_11_BSSID_LIST_EX
- struct NDIS_802_11_FIXED_IEs
- struct NDIS_802_11_WEP
- struct NDIS_802_11_KEY
- struct NDIS_802_11_REMOVE_KEY
- struct NDIS_802_11_AI_REQFI
- struct NDIS_802_11_AI_RESFI
- struct NDIS_802_11_ASSOCIATION_INFORMATION
- struct NDIS_802_11_AUTHENTICATION_ENCRYPTION
- struct NDIS_802_11_CAPABILITY
- struct BSSID_INFO
- struct NDIS_802_11_PMKID
- struct NDIS_802_11_STATUS_INDICATION
- struct PMKID_CANDIDATE
- struct NDIS_802_11_PMKID_CANDIDATE_LIST
- struct NDIS_802_11_AUTHENTICATION_REQUEST
- struct _DOT11_SCAN_REQUEST_V2

Defines

- #define **OID_802_11_BSSID** 0x0d010101
- #define **OID_802_11_SSID** 0x0d010102
- #define **OID_802_11_INFRASTRUCTURE_MODE** 0x0d010108
- #define **OID_802_11_ADD_WEP** 0x0D010113
- #define **OID_802_11_REMOVE_WEP** 0x0D010114
- #define **OID_802_11_DISASSOCIATE** 0x0D010115
- #define **OID_802_11_BSSID_LIST** 0x0d010217
- #define **OID_802_11_AUTHENTICATION_MODE** 0x0d010118
- #define **OID_802_11_PRIVACY_FILTER** 0x0d010119

- #define **OID_802_11_BSSID_LIST_SCAN** 0x0d01011A
- #define **OID_802_11_WEP_STATUS** 0x0d01011B
- #define **OID_802_11_ENCRYPTION_STATUS** OID_802_11_WEP_STATUS
- #define **OID_802_11_ADD_KEY** 0x0d01011D
- #define **OID_802_11_REMOVE_KEY** 0x0d01011E
- #define **OID_802_11_ASSOCIATION_INFORMATION** 0x0d01011F
- #define **OID_802_11_TEST** 0x0d010120
- #define **OID_802_11_CAPABILITY** 0x0d010122
- #define **OID_802_11_PMKID** 0x0d010123
- #define **NDIS_802_11_LENGTH_SSID** 32
- #define **NDIS_802_11_LENGTH_RATES** 8
- #define **NDIS_802_11_LENGTH_RATES_EX** 16
- #define **NDIS_802_11_PMKID_CANDIDATE_PREAUTH_ENABLED** 0x01
- #define **NDIS_802_11_AUTH_REQUEST_REAUTH** 0x01
- #define **NDIS_802_11_AUTH_REQUEST_KEYUPDATE** 0x02
- #define **NDIS_802_11_AUTH_REQUEST_PAIRWISE_ERROR** 0x06
- #define **NDIS_802_11_AUTH_REQUEST_GROUP_ERROR** 0x0E
- #define **OID_DOT11_NDIS_START** 0x0D010300
- #define **OID_DOT11_CURRENT_OPERATION_MODE** (OID_DOT11_NDIS_START + 8)
- #define **OID_DOT11_SCAN_REQUEST** (OID_DOT11_NDIS_START + 11)
- #define **MAX_ADAPTERS** 32

Typedefs

- typedef UCHAR **NDIS_802_11_MAC_ADDRESS** [6]
- typedef LONG **NDIS_802_11_RSSI**
- typedef enum NDIS_802_11_WEP_STATUS **NDIS_802_11_ENCRYPTION_STATUS**
- 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 ULONG **NDIS_802_11_KEY_INDEX**
- typedef ULONGLONG **NDIS_802_11_KEY_RSC**
- typedef UCHAR **NDIS_802_11_PMKID_VALUE** [16]
- typedef enum _DOT11_BSS_TYPE **DOT11_BSS_TYPE**
- typedef enum _DOT11_BSS_TYPE * **PDOT11_BSS_TYPE**
- typedef UCHAR **DOT11_MAC_ADDRESS** [6]
- typedef DOT11_MAC_ADDRESS * **PDOT11_MAC_ADDRESS**
- typedef enum _DOT11_SCAN_TYPE **DOT11_SCAN_TYPE**
- typedef enum _DOT11_SCAN_TYPE * **PDOT11_SCAN_TYPE**
- typedef struct _DOT11_SCAN_REQUEST_V2 **DOT11_SCAN_REQUEST_V2**
- typedef struct _DOT11_SCAN_REQUEST_V2 * **PDOT11_SCAN_REQUEST_V2**

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** }
- enum **_DOT11_BSS_TYPE** { **dot11_BSS_type_infrastructure** = 1, **dot11_BSS_type_-independent** = 2, **dot11_BSS_type_any** = 3 }
- enum **_DOT11_SCAN_TYPE** { **dot11_scan_type_active** = 1, **dot11_scan_type_passive** = 2, **dot11_scan_type_auto** = 3, **dot11_scan_type_forced** = 0x80000000 }

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)
- void **wpa_driver_ndis_event_adapter_arrival** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_adapter_removal** (struct wpa_driver_ndis_data *drv)

Variables

- struct wpa_driver_ops **wpa_driver_ndis_ops**

15.149.1 Detailed Description

WPA Supplicant - Windows/NDIS driver interface.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.150 src/drivers/driver_ndis.h File Reference

WPA Supplicant - Windows/NDIS driver interface.

Data Structures

- struct ndis_pmkid_entry
- struct wpa_driver_ndis_data

15.150.1 Detailed Description

WPA Supplicant - Windows/NDIS driver interface.

Copyright

15.151 src/drivers/driver_ndis_.c File Reference

WPA Supplicant - Windows/NDIS driver interface - event processing. `#include "includes.h"`

`#include "common.h"`

`#include "driver.h"`

`#include "eloop.h"`

`#include "driver_ndis.h"`

Typedefs

- typedef struct _ADAPTER * **LPADAPTER**

Enumerations

- enum **event_types** {

  EVENT_CONNECT, **EVENT_DISCONNECT**, **EVENT_MEDIA_SPECIFIC**, **EVENT_ADAPTER_ARRIVAL**,

  EVENT_ADAPTER_REMOVAL, **EVENT_CONNECT**, **EVENT_DISCONNECT**, **EVENT_MEDIA_SPECIFIC**,

  EVENT_ADAPTER_ARRIVAL, **EVENT_ADAPTER_REMOVAL** }

Functions

- void **wpa_driver_ndis_event_connect** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_disconnect** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_media_specific** (struct wpa_driver_ndis_data *drv, const u8 *data, size_t data_len)
- void **wpa_driver_ndis_event_adapter_arrival** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_adapter_removal** (struct wpa_driver_ndis_data *drv)
- void **wpa_driver_ndis_event_pipe_cb** (void *eloop_data, void *user_data)

15.151.1 Detailed Description

WPA Supplicant - Windows/NDIS driver interface - event processing.

Copyright

Copyright (c) 2004-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.152 src/drivers/driver_ndiswrapper.c File Reference

WPA Supplicant - driver interaction with Linux ndiswrapper. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
```

Data Structures

- struct wpa_driver_ndiswrapper_data
- struct wpa_key
- struct wpa_assoc_info

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

- struct wpa_driver_ops **wpa_driver_ndiswrapper_ops**

15.152.1 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 <j@w1.fi>

See README and COPYING for more details.

Please note that ndiswrapper supports WPA configuration via Linux wireless extensions and if the kernel includes support for this, driver_wext.c should be used instead of this driver wrapper.

15.152.2 Variable Documentation

15.152.2.1 struct wpa_driver_ops wpa_driver_ndiswrapper_ops

Initial value:

```
{
        .name = "ndiswrapper",
        .desc = "Linux ndiswrapper (deprecated; use wext)",
        .set_key = wpa_ndiswrapper_set_key,
        .set_countermeasures = wpa_ndiswrapper_set_countermeasures,
        .deauthenticate = wpa_ndiswrapper_deauthenticate,
        .disassociate = wpa_ndiswrapper_disassociate,
        .associate = wpa_ndiswrapper_associate,

        .get_bssid = wpa_ndiswrapper_get_bssid,
        .get_ssid = wpa_ndiswrapper_get_ssid,
        .scan2 = wpa_ndiswrapper_scan,
        .get_scan_results2 = wpa_ndiswrapper_get_scan_results,
        .init = wpa_ndiswrapper_init,
        .deinit = wpa_ndiswrapper_deinit,
        .get_capa = wpa_ndiswrapper_get_capa,
        .set_operstate = wpa_ndiswrapper_set_operstate,
}
```

15.153 src/drivers/driver_nl80211.c File Reference

Driver interaction with Linux nl80211/cfg80211. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include <net/if_arp.h>
#include <net/if.h>
#include <netlink/genl/genl.h>
#include <netlink/genl/family.h>
#include <netlink/genl/ctrl.h>
#include "nl80211_copy.h"
#include "common.h"
#include "driver.h"
#include "eloop.h"
#include "ieee802_11_defs.h"
```

Data Structures

- struct i802_bss
- struct wpa_driver_nl80211_data
- struct family_data
- struct wiphy_info_data

Defines

- #define **IFF_LOWER_UP** 0x10000
- #define **IFF_DORMANT** 0x20000
- #define **IF_OPER_DORMANT** 5
- #define **IF_OPER_UP** 6

Variables

- struct wpa_driver_ops **wpa_driver_nl80211_ops**

15.153.1 Detailed Description

Driver interaction with Linux nl80211/cfg80211.

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi> Copyright (c) 2003-2004, Instant802 Networks, Inc. Copyright (c) 2005-2006, Devicescape Software, Inc. Copyright (c) 2007, Johannes Berg <johannes@sipsolutions.net> Copyright (c) 2009, Atheros Communications

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.154 src/drivers/driver_none.c File Reference

Driver interface for RADIUS server or WPS ER only (no driver). `#include "includes.h"`

`#include "../hostapd/hostapd.h"`

`#include "driver.h"`

Data Structures

- struct none_driver_data

Variables

- struct wpa_driver_ops **wpa_driver_none_ops**

15.154.1 Detailed Description

Driver interface for RADIUS server or WPS ER only (no driver).

Copyright

Copyright (c) 2008, Atheros Communications

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.154.2 Variable Documentation

15.154.2.1 struct wpa_driver_ops wpa_driver_none_ops

Initial value:

```
{
        .name = "none",
        .desc = "no driver (RADIUS server/WPS ER)",
        .hapd_init = none_driver_hapd_init,
        .hapd_deinit = none_driver_hapd_deinit,
        .send_ether = none_driver_send_ether,
        .init = none_driver_init,
        .deinit = none_driver_deinit,
        .send_eapol = none_driver_send_eapol,
}
```

15.155 src/drivers/driver_osx.m File Reference

WPA Supplicant - Mac OS X Apple80211 driver interface. `#include "includes.h"`

```
#include <CoreFoundation/CoreFoundation.h>
#include "common.h"
#include "driver.h"
#include "eloop.h"
#include "Apple80211.h"
```

Data Structures

- struct wpa_driver_osx_data

Defines

- #define **Boolean** __DummyBoolean

Variables

- int **wpa_debug_level**
- struct wpa_driver_ops **wpa_driver_osx_ops**

15.155.1 Detailed Description

WPA Supplicant - Mac OS X Apple80211 driver interface.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.155.2 Variable Documentation

15.155.2.1 struct wpa_driver_ops wpa_driver_osx_ops

Initial value:

```
{
        .name = "osx",
        .desc = "Mac OS X Apple80211 driver",
        .get_ssid = wpa_driver_osx_get_ssid,
        .get_bssid = wpa_driver_osx_get_bssid,
        .init = wpa_driver_osx_init,
        .deinit = wpa_driver_osx_deinit,
```

```
        .scan = wpa_driver_osx_scan,
        .get_scan_results = wpa_driver_osx_get_scan_results,
        .associate = wpa_driver_osx_associate,
        .set_key = wpa_driver_osx_set_key,
        .get_capa = wpa_driver_osx_get_capa,
}
```

15.156 src/drivers/driver_prism54.c File Reference

WPA Supplicant - driver interaction with Linux Prism54.org driver. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "driver_wext.h"
#include "driver_hostap.h"
```

Data Structures

- struct wpa_driver_prism54_data

Defines

- #define **PRISM54_SET_WPA** SIOCIWFIRSTPRIV+12
- #define **PRISM54_HOSTAPD** SIOCIWFIRSTPRIV+25
- #define **PRISM54_DROP_UNENCRYPTED** SIOCIWFIRSTPRIV+26

Variables

- struct wpa_driver_ops **wpa_driver_prism54_ops**

15.156.1 Detailed Description

WPA Supplicant - driver interaction with Linux Prism54.org driver.

Copyright

Copyright (c) 2002-2007, Jouni Malinen <j@w1.fi> Copyright (c) 2004, Luis R. Rodriguez <mcgrof@ruslug.rutgers.edu> Copyright (c) 2004, Bell Kin <bell_-kin@pek.com.tw>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.157 src/drivers/driver_privsep.c File Reference

WPA Supplicant - privilege separated driver interface. `#include "includes.h"`

```
#include <sys/un.h>
#include "common.h"
#include "driver.h"
#include "eloop.h"
#include "privsep_commands.h"
```

Data Structures

- struct wpa_driver_privsep_data

Variables

- struct wpa_driver_ops **wpa_driver_privsep_ops**
- struct wpa_driver_ops * **wpa_drivers** [ ]

15.157.1 Detailed Description

WPA Supplicant - privilege separated driver interface.

Copyright

Copyright (c) 2007-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.157.2 Variable Documentation

15.157.2.1 struct wpa_driver_ops wpa_driver_privsep_ops

Initial value:

```
{
        "privsep",
        "wpa_supplicant privilege separated driver",
        .get_bssid = wpa_driver_privsep_get_bssid,
        .get_ssid = wpa_driver_privsep_get_ssid,
        .set_key = wpa_driver_privsep_set_key,
        .init = wpa_driver_privsep_init,
        .deinit = wpa_driver_privsep_deinit,
        .set_param = wpa_driver_privsep_set_param,
        .scan2 = wpa_driver_privsep_scan,
        .deauthenticate = wpa_driver_privsep_deauthenticate,
        .disassociate = wpa_driver_privsep_disassociate,
        .associate = wpa_driver_privsep_associate,
```

```
        .get_capa = wpa_driver_privsep_get_capa,
        .get_mac_addr = wpa_driver_privsep_get_mac_addr,
        .get_scan_results2 = wpa_driver_privsep_get_scan_results2,
        .set_country = wpa_driver_privsep_set_country,
}
```

15.157.2.2 struct wpa_driver_ops* wpa_drivers[ ]

Initial value:

```
{
        &wpa_driver_privsep_ops,
        NULL
}
```

15.158 src/drivers/driver_ps3.c File Reference

WPA Supplicant - PS3 Linux wireless extension driver interface. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "wpa_common.h"
#include "driver.h"
#include "eloop.h"
#include "driver_wext.h"
#include "ieee802_11_defs.h"
```

Variables

- struct wpa_driver_ops **wpa_driver_ps3_ops**

15.158.1 Detailed Description

WPA Supplicant - PS3 Linux wireless extension driver interface.

Copyright

Copyright 2007, 2008 Sony Corporation

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.158.2 Variable Documentation

15.158.2.1 struct wpa_driver_ops wpa_driver_ps3_ops

Initial value:

```
{
        .name = "ps3",
        .desc = "PLAYSTATION3 Linux wireless extension driver",
        .get_bssid = wpa_driver_wext_get_bssid,
        .get_ssid = wpa_driver_wext_get_ssid,
        .scan2 = wpa_driver_wext_scan,
        .get_scan_results2 = wpa_driver_wext_get_scan_results,
        .associate = wpa_driver_ps3_associate,
        .init = wpa_driver_wext_init,
        .deinit = wpa_driver_wext_deinit,
        .get_capa = wpa_driver_ps3_get_capa,
}
```

15.159 src/drivers/driver_ralink.c File Reference

WPA Supplicant - driver interaction with Ralink Wireless Client. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "l2_packet/l2_packet.h"
#include "eloop.h"
#include "ieee802_11_defs.h"
#include "priv_netlink.h"
#include "driver_ralink.h"
```

Data Structures

- struct wpa_driver_ralink_data

Defines

- #define **MAX_SSID_LEN** 32

Variables

- struct wpa_driver_ops **wpa_driver_ralink_ops**

15.159.1 Detailed Description

WPA Supplicant - driver interaction with Ralink Wireless Client.

Copyright

> Copyright (c) 2003-2006, Jouni Malinen <j@w1.fi> Copyright (c) 2007, Snowpin Lee <snowpin_lee@ralinktech.com.tw>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.159.2 Variable Documentation

15.159.2.1 struct wpa_driver_ops wpa_driver_ralink_ops

Initial value:

```
{
        .name = "ralink",
        .desc = "Ralink Wireless Client driver",
        .get_bssid = wpa_driver_ralink_get_bssid,
        .get_ssid = wpa_driver_ralink_get_ssid,
        .set_key = wpa_driver_ralink_set_key,
        .init = wpa_driver_ralink_init,
        .deinit = wpa_driver_ralink_deinit,
        .set_countermeasures    = wpa_driver_ralink_set_countermeasures,
        .scan2 = wpa_driver_ralink_scan,
        .get_scan_results2 = wpa_driver_ralink_get_scan_results,
        .deauthenticate = wpa_driver_ralink_deauthenticate,
        .disassociate = wpa_driver_ralink_disassociate,
        .associate = wpa_driver_ralink_associate,
        .add_pmkid = wpa_driver_ralink_add_pmkid,
        .remove_pmkid = wpa_driver_ralink_remove_pmkid,
        .flush_pmkid = wpa_driver_ralink_flush_pmkid,
}
```

15.160 src/drivers/driver_ralink.h File Reference

WPA Supplicant - driver_ralink exported functions.

Data Structures

- union _LARGE_INTEGER
- struct _NDIS_802_11_CONFIGURATION_FH
- struct _NDIS_802_11_CONFIGURATION
- struct _NDIS_802_11_KEY
- struct _NDIS_802_11_REMOVE_KEY
- struct _NDIS_802_11_WEP
- struct _BSSID_INFO
- struct _NDIS_802_11_PMKID
- struct _PMKID_CANDIDATE
- struct _NDIS_802_11_PMKID_CANDIDATE_LIST
- struct _NDIS_802_11_SSID
- struct _NDIS_WLAN_BSSID
- struct _NDIS_802_11_BSSID_LIST
- struct _NDIS_WLAN_BSSID_EX
- struct _NDIS_802_11_BSSID_LIST_EX
- struct _NDIS_802_11_FIXED_IEs
- struct _NDIS_802_11_AI_REQFI
- struct _NDIS_802_11_AI_RESFI
- struct _NDIS_802_11_ASSOCIATION_INFORMATION
- struct ndis_pmkid_entry
- struct _MLME_DEAUTH_REQ_STRUCT

Defines

- #define **RT_PRIV_IOCTL** (SIOCIWFIRSTPRIV + 0x0E)
- #define **RTPRIV_IOCTL_SET** (SIOCIWFIRSTPRIV + 0x02)
- #define **OID_GET_SET_TOGGLE** 0x8000
- #define **OID_802_11_ADD_WEP** 0x0112
- #define **OID_802_11_REMOVE_WEP** 0x0113
- #define **OID_802_11_DISASSOCIATE** 0x0114
- #define **OID_802_11_PRIVACY_FILTER** 0x0118
- #define **OID_802_11_ASSOCIATION_INFORMATION** 0x011E
- #define **OID_802_11_BSSID_LIST_SCAN** 0x0508
- #define **OID_802_11_SSID** 0x0509
- #define **OID_802_11_BSSID** 0x050A
- #define **OID_802_11_WEP_STATUS** 0x0510
- #define **OID_802_11_AUTHENTICATION_MODE** 0x0511
- #define **OID_802_11_INFRASTRUCTURE_MODE** 0x0512
- #define **OID_802_11_TX_POWER_LEVEL** 0x0517
- #define **OID_802_11_REMOVE_KEY** 0x0519
- #define **OID_802_11_ADD_KEY** 0x0520
- #define **OID_802_11_DEAUTHENTICATION** 0x0526
- #define **OID_802_11_DROP_UNENCRYPTED** 0x0527

- #define **OID_802_11_BSSID_LIST** 0x0609
- #define **OID_802_3_CURRENT_ADDRESS** 0x060A
- #define **OID_SET_COUNTERMEASURES** 0x0616
- #define **OID_802_11_SET_IEEE8021X** 0x0617
- #define **OID_802_11_SET_IEEE8021X_REQUIRE_KEY** 0x0618
- #define **OID_802_11_PMKID** 0x0620
- #define **RT_OID_WPA_SUPPLICANT_SUPPORT** 0x0621
- #define **RT_OID_WE_VERSION_COMPILED** 0x0622
- #define **RT_OID_NEW_DRIVER** 0x0623
- #define **PACKED** __attribute__ ((packed))
- #define **RT_ASSOC_EVENT_FLAG** 0x0101
- #define **RT_DISASSOC_EVENT_FLAG** 0x0102
- #define **RT_REQIE_EVENT_FLAG** 0x0103
- #define **RT_RESPIE_EVENT_FLAG** 0x0104
- #define **RT_ASSOCINFO_EVENT_FLAG** 0x0105
- #define **RT_PMKIDCAND_FLAG** 0x0106
- #define **RT_INTERFACE_DOWN** 0x0107
- #define **RT_REPORT_AP_INFO** 0x0108
- #define **CHAR** char
- #define **INT** int
- #define **SHORT** int
- #define **UINT** u32
- #define **ULONG** unsigned long
- #define **USHORT** unsigned short
- #define **UCHAR** unsigned char
- #define **uint32** u32
- #define **uint8** u8
- #define **BOOLEAN** u8
- #define **VOID** void
- #define **LONG** long
- #define **LONGLONG** s64
- #define **ULONGLONG** u64
- #define **NDIS_802_11_LENGTH_SSID** 32
- #define **NDIS_802_11_LENGTH_RATES** 8
- #define **NDIS_802_11_LENGTH_RATES_EX** 16
- #define **MAX_LEN_OF_SSID** 32
- #define **MAC_ADDR_LEN** 6
- #define **NDIS_802_11_AUTH_REQUEST_AUTH_FIELDS** 0x0f
- #define **NDIS_802_11_AUTH_REQUEST_REAUTH** 0x01
- #define **NDIS_802_11_AUTH_REQUEST_KEYUPDATE** 0x02
- #define **NDIS_802_11_AUTH_REQUEST_PAIRWISE_ERROR** 0x06
- #define **NDIS_802_11_AUTH_REQUEST_GROUP_ERROR** 0x0E
- #define **NDIS_802_11_PMKID_CANDIDATE_PREAUTH_ENABLED** 0x01
- #define **NDIS_802_11_AI_REQFI_CAPABILITIES** 1
- #define **NDIS_802_11_AI_REQFI_LISTENINTERVAL** 2
- #define **NDIS_802_11_AI_REQFI_CURRENTAPADDRESS** 4
- #define **NDIS_802_11_AI_RESFI_CAPABILITIES** 1
- #define **NDIS_802_11_AI_RESFI_STATUSCODE** 2
- #define **NDIS_802_11_AI_RESFI_ASSOCIATIONID** 4

Typedefs

- typedef VOID * **PVOID**
- typedef CHAR * **PCHAR**
- typedef UCHAR * **PUCHAR**
- typedef USHORT * **PUSHORT**
- typedef LONG * **PLONG**
- typedef ULONG * **PULONG**
- typedef union _LARGE_INTEGER **LARGE_INTEGER**
- typedef UCHAR **NDIS_802_11_MAC_ADDRESS** [6]
- typedef enum _NDIS_802_11_NETWORK_TYPE **NDIS_802_11_NETWORK_TYPE**
- typedef enum _NDIS_802_11_NETWORK_TYPE * **PNDIS_802_11_NETWORK_TYPE**
- typedef LONG **NDIS_802_11_RSSI**
- typedef struct _NDIS_802_11_CONFIGURATION_FH **NDIS_802_11_CONFIGURATION_FH**
- typedef struct _NDIS_802_11_CONFIGURATION_FH * **PNDIS_802_11_CONFIGURATION_FH**
- typedef struct _NDIS_802_11_CONFIGURATION **NDIS_802_11_CONFIGURATION**
- typedef struct _NDIS_802_11_CONFIGURATION * **PNDIS_802_11_CONFIGURATION**
- typedef ULONG **NDIS_802_11_KEY_INDEX**
- typedef ULONGLONG **NDIS_802_11_KEY_RSC**
- typedef struct _NDIS_802_11_KEY **NDIS_802_11_KEY**
- typedef struct _NDIS_802_11_KEY * **PNDIS_802_11_KEY**
- typedef struct _NDIS_802_11_REMOVE_KEY **NDIS_802_11_REMOVE_KEY**
- typedef struct _NDIS_802_11_REMOVE_KEY * **PNDIS_802_11_REMOVE_KEY**
- typedef struct PACKED _NDIS_802_11_WEP **NDIS_802_11_WEP**
- typedef struct PACKED _NDIS_802_11_WEP * **PNDIS_802_11_WEP**
- typedef enum _NDIS_802_11_NETWORK_INFRASTRUCTURE **NDIS_802_11_NETWORK_INFRASTRUCTURE**
- typedef enum _NDIS_802_11_NETWORK_INFRASTRUCTURE * **PNDIS_802_11_NETWORK_INFRASTRUCTURE**
- typedef UCHAR **NDIS_802_11_PMKID_VALUE** [16]
- typedef struct _BSSID_INFO **BSSID_INFO**
- typedef struct _BSSID_INFO * **PBSSID_INFO**
- typedef struct _NDIS_802_11_PMKID **NDIS_802_11_PMKID**
- typedef struct _NDIS_802_11_PMKID * **PNDIS_802_11_PMKID**
- typedef struct _PMKID_CANDIDATE **PMKID_CANDIDATE**
- typedef struct _PMKID_CANDIDATE * **PPMKID_CANDIDATE**
- typedef struct _NDIS_802_11_PMKID_CANDIDATE_LIST **NDIS_802_11_PMKID_CANDIDATE_LIST**
- typedef struct _NDIS_802_11_PMKID_CANDIDATE_LIST * **PNDIS_802_11_PMKID_CANDIDATE_LIST**
- typedef enum _NDIS_802_11_AUTHENTICATION_MODE **NDIS_802_11_AUTHENTICATION_MODE**
- typedef enum _NDIS_802_11_AUTHENTICATION_MODE * **PNDIS_802_11_AUTHENTICATION_MODE**
- 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 struct PACKED _NDIS_802_11_SSID **NDIS_802_11_SSID**
- typedef struct PACKED _NDIS_802_11_SSID * **PNDIS_802_11_SSID**
- typedef struct PACKED _NDIS_WLAN_BSSID **NDIS_WLAN_BSSID**
- typedef struct PACKED _NDIS_WLAN_BSSID * **PNDIS_WLAN_BSSID**

- typedef struct PACKED _NDIS_802_11_BSSID_LIST **NDIS_802_11_BSSID_LIST**
- typedef struct PACKED _NDIS_802_11_BSSID_LIST * **PNDIS_802_11_BSSID_LIST**
- typedef struct PACKED _NDIS_WLAN_BSSID_EX **NDIS_WLAN_BSSID_EX**
- typedef struct PACKED _NDIS_WLAN_BSSID_EX * **PNDIS_WLAN_BSSID_EX**
- typedef struct PACKED _NDIS_802_11_BSSID_LIST_EX **NDIS_802_11_BSSID_LIST_EX**
- typedef struct PACKED _NDIS_802_11_BSSID_LIST_EX * **PNDIS_802_11_BSSID_LIST_EX**
- typedef struct PACKED _NDIS_802_11_FIXED_IEs **NDIS_802_11_FIXED_IEs**
- typedef struct PACKED _NDIS_802_11_FIXED_IEs * **PNDIS_802_11_FIXED_IEs**
- typedef enum _NDIS_802_11_WEP_STATUS **NDIS_802_11_WEP_STATUS**
- typedef enum _NDIS_802_11_WEP_STATUS * **PNDIS_802_11_WEP_STATUS**
- typedef enum _NDIS_802_11_WEP_STATUS **NDIS_802_11_ENCRYPTION_STATUS**
- typedef enum _NDIS_802_11_WEP_STATUS * **PNDIS_802_11_ENCRYPTION_STATUS**
- typedef enum _NDIS_802_11_RELOAD_DEFAULTS **NDIS_802_11_RELOAD_DEFAULTS**
- typedef enum _NDIS_802_11_RELOAD_DEFAULTS * **PNDIS_802_11_RELOAD_DEFAULTS**
- typedef struct _NDIS_802_11_AI_REQFI **NDIS_802_11_AI_REQFI**
- typedef struct _NDIS_802_11_AI_REQFI * **PNDIS_802_11_AI_REQFI**
- typedef struct _NDIS_802_11_AI_RESFI **NDIS_802_11_AI_RESFI**
- typedef struct _NDIS_802_11_AI_RESFI * **PNDIS_802_11_AI_RESFI**
- typedef struct _NDIS_802_11_ASSOCIATION_INFORMATION **NDIS_802_11_-ASSOCIATION_INFORMATION**
- typedef struct _NDIS_802_11_ASSOCIATION_INFORMATION * **PNDIS_802_11_-ASSOCIATION_INFORMATION**
- typedef struct _MLME_DEAUTH_REQ_STRUCT **MLME_DEAUTH_REQ_STRUCT**
- typedef struct _MLME_DEAUTH_REQ_STRUCT * **PMLME_DEAUTH_REQ_STRUCT**

Enumerations

- enum **_NDIS_802_11_NETWORK_TYPE** {

  Ndis802_11FH, **Ndis802_11DS**, **Ndis802_11OFDM5**, **Ndis802_11OFDM24**,

  Ndis802_11Automode, **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_RELOAD_DEFAULTS** { **Ndis802_11ReloadWEPKeys** }

15.160.1 Detailed Description

WPA Supplicant - driver_ralink exported functions.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi> Copyright (c) 2007, Snowpin Lee <snowpin_lee@ralinktech.com.tw>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.161 src/drivers/driver_roboswitch.c File Reference

WPA Supplicant - roboswitch driver interface. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include <linux/if.h>
#include <linux/sockios.h>
#include <linux/if_ether.h>
#include <linux/mii.h>
#include "common.h"
#include "driver.h"
#include "l2_packet/l2_packet.h"
```

Data Structures

- struct wpa_driver_roboswitch_data

Defines

- #define **ROBO_PHY_ADDR** 0x1e
- #define **ROBO_MII_PAGE** 0x10
- #define **ROBO_MII_ADDR** 0x11
- #define **ROBO_MII_DATA_OFFSET** 0x18
- #define **ROBO_MII_PAGE_ENABLE** 0x01
- #define **ROBO_MII_ADDR_WRITE** 0x01
- #define **ROBO_MII_ADDR_READ** 0x02
- #define **ROBO_MII_DATA_MAX** 4
- #define **ROBO_MII_RETRY_MAX** 10
- #define **ROBO_ARLCTRL_PAGE** 0x04
- #define **ROBO_VLAN_PAGE** 0x34
- #define **ROBO_ARLCTRL_CONF** 0x00
- #define **ROBO_ARLCTRL_ADDR_1** 0x10
- #define **ROBO_ARLCTRL_VEC_1** 0x16
- #define **ROBO_ARLCTRL_ADDR_2** 0x20
- #define **ROBO_ARLCTRL_VEC_2** 0x26
- #define **ROBO_VLAN_ACCESS** 0x08
- #define **ROBO_VLAN_ACCESS_5350** 0x06
- #define **ROBO_VLAN_READ** 0x0c
- #define **ROBO_VLAN_MAX** 0xff

Variables

- struct wpa_driver_ops **wpa_driver_roboswitch_ops**

15.161.1 Detailed Description

WPA Supplicant - roboswitch driver interface.

Copyright

Copyright (c) 2008-2009 Jouke Witteveen

15.161.2 Variable Documentation

15.161.2.1 struct wpa_driver_ops wpa_driver_roboswitch_ops

Initial value:

```
{
        .name = "roboswitch",
        .desc = "wpa_supplicant roboswitch driver",
        .get_ssid = wpa_driver_roboswitch_get_ssid,
        .get_bssid = wpa_driver_roboswitch_get_bssid,
        .get_capa = wpa_driver_roboswitch_get_capa,
        .init = wpa_driver_roboswitch_init,
        .deinit = wpa_driver_roboswitch_deinit,
        .set_param = wpa_driver_roboswitch_set_param,
        .get_ifname = wpa_driver_roboswitch_get_ifname,
}
```

15.162 src/drivers/driver_test.c File Reference

WPA Supplicant - testing driver interface. `#include "build_config.h"`

```
#include "includes.h"
#include <sys/un.h>
#include <dirent.h>
#include <sys/stat.h>
#include "common.h"
#include "driver.h"
#include "l2_packet/l2_packet.h"
#include "eloop.h"
#include "sha1.h"
#include "ieee802_11_defs.h"
#include "../../hostapd/hostapd.h"
#include "../../hostapd/wpa.h"
#include "../../hostapd/hw_features.h"
```

Data Structures

- struct test_client_socket
- struct test_driver_bss
- struct wpa_driver_test_global
- struct wpa_driver_test_data

Defines

- #define **DRIVER_TEST_UNIX**
- #define **MAX_SCAN_RESULTS** 30
- #define **MAX_IE_LEN** 1000

Variables

- struct wpa_driver_ops **wpa_driver_test_ops**

15.162.1 Detailed Description

WPA Supplicant - testing driver interface.

Copyright

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.163 src/drivers/driver_wext.c File Reference

WPA Supplicant - driver interaction with generic Linux Wireless Extensions. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include <net/if_arp.h>
#include "wireless_copy.h"
#include "common.h"
#include "driver.h"
#include "eloop.h"
#include "priv_netlink.h"
#include "driver_wext.h"
#include "ieee802_11_defs.h"
#include "wpa_common.h"
```

Data Structures

- struct wext_scan_data

Functions

- int **wpa_driver_wext_set_auth_param** (struct wpa_driver_wext_data *drv, int idx, u32 value)
- int wpa_driver_wext_get_bssid (void *priv, u8 *bssid)

  Get BSSID, SIOCGIWAP.

- int wpa_driver_wext_set_bssid (void *priv, const u8 *bssid)

  Set BSSID, SIOCSIWAP.

- int wpa_driver_wext_get_ssid (void *priv, u8 *ssid)

  Get SSID, SIOCGIWESSID.

- int wpa_driver_wext_set_ssid (void *priv, const u8 *ssid, size_t ssid_len)

  Set SSID, SIOCSIWESSID.

- int wpa_driver_wext_set_freq (void *priv, int freq)

  Set frequency/channel, SIOCSIWFREQ.

- int wpa_driver_wext_get_ifflags (struct wpa_driver_wext_data *drv, int *flags)

  Get interface flags (SIOCGIFFLAGS).

- int wpa_driver_wext_set_ifflags (struct wpa_driver_wext_data *drv, int flags)

  Set interface flags (SIOCSIFFLAGS).

- void * wpa_driver_wext_init (void *ctx, const char *ifname)

  Initialize WE driver interface.

- void wpa_driver_wext_deinit (void *priv)

  Deinitialize WE driver interface.

- void wpa_driver_wext_scan_timeout (void *eloop_ctx, void *timeout_ctx)

  Scan timeout to report scan completion.

- int wpa_driver_wext_scan (void *priv, struct wpa_driver_scan_params *params)

  Request the driver to initiate scan.

- struct wpa_scan_results * wpa_driver_wext_get_scan_results (void *priv)

  Fetch the latest scan results.

- int wpa_driver_wext_set_key (const char *ifname, 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_cipher2wext** (int cipher)
- int **wpa_driver_wext_keymgmt2wext** (int keymgmt)
- int **wpa_driver_wext_associate** (void *priv, struct wpa_driver_associate_params *params)
- int wpa_driver_wext_set_mode (void *priv, int mode)

  Set wireless mode (infra/adhoc), SIOCSIWMODE.

- int **wpa_driver_wext_get_capa** (void *priv, struct wpa_driver_capa *capa)
- int **wpa_driver_wext_alternative_ifindex** (struct wpa_driver_wext_data *drv, const char *ifname)
- int **wpa_driver_wext_set_operstate** (void *priv, int state)
- int **wpa_driver_wext_get_version** (struct wpa_driver_wext_data *drv)

Variables

- struct wpa_driver_ops **wpa_driver_wext_ops**

15.163.1 Detailed Description

WPA Supplicant - driver interaction with generic Linux Wireless Extensions.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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.

15.163.2 Function Documentation

15.163.2.1 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().

15.163.2.2 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

15.163.2.3 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

15.163.2.4 struct wpa_scan_results* wpa_driver_wext_get_scan_results (void * *priv*) [read]

Fetch the latest scan results.

Parameters:

priv Pointer to private wext data from wpa_driver_wext_init()

Returns:

Scan results on success, -1 on failure

15.163.2.5 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

15.163.2.6 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

15.163.2.7 int wpa_driver_wext_scan (void * *priv*, struct wpa_driver_scan_params * *params*)

Request the driver to initiate scan.

Parameters:

priv Pointer to private wext data from wpa_driver_wext_init()

param Scan parameters (specific SSID to scan for (ProbeReq), etc.)

Returns:

0 on success, -1 on failure

15.163.2.8 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.

15.163.2.9 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

15.163.2.10 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

15.163.2.11 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

15.163.2.12 int wpa_driver_wext_set_key (const char * *ifname*, void * *priv*, wpa_alg *alg*, const u8 * *addr*, int *key_idx*, int *set_tx*, const u8 * *seq*, size_t *seq_len*, const u8 * *key*, size_t *key_len*)

Configure encryption key.

Parameters:

priv Pointer to private wext data from wpa_driver_wext_init()

priv Private driver interface data

alg Encryption algorithm (WPA_ALG_NONE, WPA_ALG_WEP, WPA_ALG_TKIP, WPA_ALG_-CCMP); WPA_ALG_NONE clears the key.

addr Address of the peer STA or ff:ff:ff:ff:ff:ff for broadcast/default keys

key_idx key index (0..3), usually 0 for unicast keys

set_tx Configure this key as the default Tx key (only used when driver does not support separate unicast/individual key

seq Sequence number/packet number, seq_len octets, the next packet number to be used for in replay protection; configured for Rx keys (in most cases, this is only used with broadcast keys and set to zero for unicast keys)

seq_len Length of the seq, depends on the algorithm: TKIP: 6 octets, CCMP: 6 octets

key Key buffer; TKIP: 16-byte temporal key, 8-byte Tx Mic key, 8-byte Rx Mic Key

key_len Length of the key buffer in octets (WEP: 5 or 13, TKIP: 32, CCMP: 16)

Returns:

0 on success, -1 on failure

This function uses SIOCSIWENCODEEXT by default, but tries to use SIOCSIWENCODE if the extended ioctl fails when configuring a WEP key.

15.163.2.13 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

15.163.2.14 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

15.163.3 Variable Documentation

15.163.3.1 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_key = wpa_driver_wext_set_key,
        .set_countermeasures = wpa_driver_wext_set_countermeasures,
        .scan2 = wpa_driver_wext_scan,
        .get_scan_results2 = wpa_driver_wext_get_scan_results,
        .deauthenticate = wpa_driver_wext_deauthenticate,
        .disassociate = wpa_driver_wext_disassociate,
        .associate = wpa_driver_wext_associate,
        .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,
        .set_operstate = wpa_driver_wext_set_operstate,
}
```

15.164 src/drivers/driver_wext.h File Reference

WPA Supplicant - driver_wext exported functions. `#include <net/if.h>`

Data Structures

- struct wpa_driver_wext_data

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 (const char *ifname, 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, struct wpa_driver_scan_params *params)
  Request the driver to initiate scan.
- struct wpa_scan_results * wpa_driver_wext_get_scan_results (void *priv)
  Fetch the latest scan results.
- void wpa_driver_wext_scan_timeout (void *eloop_ctx, void *timeout_ctx)
  Scan timeout to report scan completion.
- int **wpa_driver_wext_alternative_ifindex** (struct wpa_driver_wext_data *drv, const char *ifname)
- void * wpa_driver_wext_init (void *ctx, const char *ifname)

Initialize WE driver interface.

- void wpa_driver_wext_deinit (void *priv)
  Deinitialize WE driver interface.

- int **wpa_driver_wext_set_operstate** (void *priv, int state)
- int **wpa_driver_wext_get_version** (struct wpa_driver_wext_data *drv)
- int **wpa_driver_wext_associate** (void *priv, struct wpa_driver_associate_params *params)
- int **wpa_driver_wext_get_capa** (void *priv, struct wpa_driver_capa *capa)
- int **wpa_driver_wext_set_auth_param** (struct wpa_driver_wext_data *drv, int idx, u32 value)
- int **wpa_driver_wext_cipher2wext** (int cipher)
- int **wpa_driver_wext_keymgmt2wext** (int keymgmt)

15.164.1 Detailed Description

WPA Supplicant - driver_wext exported functions.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.164.2 Function Documentation

15.164.2.1 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().

15.164.2.2 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

15.164.2.3 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

15.164.2.4 struct wpa_scan_results* wpa_driver_wext_get_scan_results (void * *priv*) [read]

Fetch the latest scan results.

Parameters:

priv Pointer to private wext data from wpa_driver_wext_init()

Returns:

Scan results on success, -1 on failure

15.164.2.5 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

15.164.2.6 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

15.164.2.7 int wpa_driver_wext_scan (void * *priv*, struct wpa_driver_scan_params * *params*)

Request the driver to initiate scan.

Parameters:

priv Pointer to private wext data from wpa_driver_wext_init()

param Scan parameters (specific SSID to scan for (ProbeReq), etc.)

Returns:

0 on success, -1 on failure

15.164.2.8 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.

15.164.2.9 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

15.164.2.10 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

15.164.2.11 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

15.164.2.12 int wpa_driver_wext_set_key (const char * *ifname*, void * *priv*, wpa_alg *alg*, const u8 * *addr*, int *key_idx*, int *set_tx*, const u8 * *seq*, size_t *seq_len*, const u8 * *key*, size_t *key_len*)

Configure encryption key.

Parameters:

priv Pointer to private wext data from wpa_driver_wext_init()

priv Private driver interface data

alg Encryption algorithm (WPA_ALG_NONE, WPA_ALG_WEP, WPA_ALG_TKIP, WPA_ALG_-CCMP); WPA_ALG_NONE clears the key.

addr Address of the peer STA or ff:ff:ff:ff:ff:ff for broadcast/default keys

key_idx key index (0..3), usually 0 for unicast keys

set_tx Configure this key as the default Tx key (only used when driver does not support separate unicast/individual key

seq Sequence number/packet number, seq_len octets, the next packet number to be used for in replay protection; configured for Rx keys (in most cases, this is only used with broadcast keys and set to zero for unicast keys)

seq_len Length of the seq, depends on the algorithm: TKIP: 6 octets, CCMP: 6 octets

key Key buffer; TKIP: 16-byte temporal key, 8-byte Tx Mic key, 8-byte Rx Mic Key

key_len Length of the key buffer in octets (WEP: 5 or 13, TKIP: 32, CCMP: 16)

Returns:

0 on success, -1 on failure

This function uses SIOCSIWENCODEEXT by default, but tries to use SIOCSIWENCODE if the extended ioctl fails when configuring a WEP key.

15.164.2.13 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

15.164.2.14 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

15.165 src/drivers/driver_wired.c File Reference

WPA Supplicant - wired Ethernet driver interface. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include <net/if.h>
#include "common.h"
#include "driver.h"
```

Data Structures

- struct ieee8023_hdr
- struct wpa_driver_wired_data

Variables

- struct ieee8023_hdr **STRUCT_PACKED**
- struct wpa_driver_ops **wpa_driver_wired_ops**

15.165.1 Detailed Description

WPA Supplicant - wired Ethernet driver interface.

Copyright

> Copyright (c) 2005-2007, Jouni Malinen <j@w1.fi> Copyright (c) 2004, Gunter Burchardt <tira@isx.de>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.165.2 Variable Documentation

15.165.2.1 struct wpa_driver_ops wpa_driver_wired_ops

Initial value:

```
{
        .name = "wired",
        .desc = "Wired Ethernet driver",




        .get_ssid = wpa_driver_wired_get_ssid,
        .get_bssid = wpa_driver_wired_get_bssid,
        .get_capa = wpa_driver_wired_get_capa,
        .init = wpa_driver_wired_init,
```

```
        .deinit = wpa_driver_wired_deinit,

}
```

15.166 src/drivers/drivers.c File Reference

Driver interface list. `#include "includes.h"`

Variables

- struct wpa_driver_ops * **wpa_drivers** [ ]

15.166.1 Detailed Description

Driver interface list.

Copyright

Copyright (c) 2004-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.167 src/drivers/ndis_events.c File Reference

ndis_events - Receive NdisMIndicateStatus() events using WMI `#include "includes.h"`

```
#include <wbemidl.h>
#include "common.h"
```

Data Structures

- struct ndis_events_data

Defines

- #define **_WIN32_WINNT** 0x0400
- #define **BstrAlloc**(x) (x) ? SysAllocString(x) : NULL
- #define **BstrFree**(x) if (x) SysFreeString(x)
- #define **MAX_QUERY_LEN** 256

Enumerations

- enum **event_types** {

  EVENT_CONNECT, **EVENT_DISCONNECT**, **EVENT_MEDIA_SPECIFIC**, **EVENT_-ADAPTER_ARRIVAL**,

  EVENT_ADAPTER_REMOVAL, **EVENT_CONNECT**, **EVENT_DISCONNECT**, **EVENT_-MEDIA_SPECIFIC**,

  EVENT_ADAPTER_ARRIVAL, **EVENT_ADAPTER_REMOVAL** }

Functions

- HRESULT STDMETHODCALLTYPE **call_IWbemServices_ExecQuery** (IWbemServices *pSvc, LPCWSTR strQueryLanguage, LPCWSTR strQuery, long lFlags, IWbemContext *pCtx, IEnumWbemClassObject **ppEnum)
- HRESULT STDMETHODCALLTYPE **call_IWbemServices_ExecNotificationQueryAsync** (IWbemServices *pSvc, LPCWSTR strQueryLanguage, LPCWSTR strQuery, long lFlags, IWbemContext *pCtx, IWbemObjectSink *pResponseHandler)
- HRESULT STDMETHODCALLTYPE **call_IWbemLocator_ConnectServer** (IWbemLocator *pLoc, LPCWSTR strNetworkResource, LPCWSTR strUser, LPCWSTR strPassword, LPCWSTR strLocale, long lSecurityFlags, LPCWSTR strAuthority, IWbemContext *pCtx, IWbemServices **ppNamespace)
- void **ndis_events_deinit** (struct ndis_events_data *events)
- struct ndis_events_data * **ndis_events_init** (HANDLE *read_pipe, HANDLE *event_avail, const char *ifname, const char *desc)

15.167.1 Detailed Description

ndis_events - Receive NdisMIndicateStatus() events using WMI

Copyright

15.168 src/drivers/priv_netlink.h File Reference

wpa_supplicant - Private copy of Linux netlink/rtnetlink definitions.

Data Structures

- struct sockaddr_nl
- struct nlmsghdr
- struct ifinfomsg
- struct rtattr

Defines

- #define **IFLA_IFNAME** 3
- #define **IFLA_WIRELESS** 11
- #define **IFLA_OPERSTATE** 16
- #define **IFLA_LINKMODE** 17
- #define **IF_OPER_DORMANT** 5
- #define **IF_OPER_UP** 6
- #define **NLM_F_REQUEST** 1
- #define **NETLINK_ROUTE** 0
- #define **RTMGRP_LINK** 1
- #define **RTM_BASE** 0x10
- #define **RTM_NEWLINK** (RTM_BASE + 0)
- #define **RTM_DELLINK** (RTM_BASE + 1)
- #define **RTM_SETLINK** (RTM_BASE + 3)
- #define **NLMSG_ALIGNTO** 4
- #define **NLMSG_ALIGN**(len) (((len) + NLMSG_ALIGNTO - 1) & ~(NLMSG_ALIGNTO - 1))
- #define **NLMSG_LENGTH**(len) ((len) + NLMSG_ALIGN(sizeof(struct nlmsghdr)))
- #define **NLMSG_DATA**(nlh) ((void*) (((char*) nlh) + NLMSG_LENGTH(0)))
- #define **RTA_ALIGNTO** 4
- #define **RTA_ALIGN**(len) (((len) + RTA_ALIGNTO - 1) & ~(RTA_ALIGNTO - 1))
- #define **RTA_OK**(rta, len)
- #define **RTA_NEXT**(rta, attrlen)
- #define **RTA_LENGTH**(len) (RTA_ALIGN(sizeof(struct rtattr)) + (len))
- #define **RTA_DATA**(rta) ((void *) (((char *) (rta)) + RTA_LENGTH(0)))

15.168.1 Detailed Description

wpa_supplicant - Private copy of Linux netlink/rtnetlink definitions.

Copyright

15.168.2 Define Documentation

15.168.2.1 #define RTA_NEXT(rta, attrlen)

Value:

```
((attrlen) -= RTA_ALIGN((rta)->rta_len), \
(struct rtattr *) (((char *)(rta)) + RTA_ALIGN((rta)->rta_len)))
```

15.168.2.2 #define RTA_OK(rta, len)

Value:

```
((len) > 0 && (rta)->rta_len >= sizeof(struct rtattr) && \
(rta)->rta_len <= (len))
```

15.169 src/drivers/scan_helpers.c File Reference

WPA Supplicant - Helper functions for scan result processing. `#include "includes.h"`

`#include "common.h"`

`#include "drivers/driver.h"`

`#include "ieee802_11_defs.h"`

Functions

- const u8 * **wpa_scan_get_ie** (const struct wpa_scan_res *res, u8 ie)
- const u8 * **wpa_scan_get_vendor_ie** (const struct wpa_scan_res *res, u32 vendor_type)
- struct wpabuf * **wpa_scan_get_vendor_ie_multi** (const struct wpa_scan_res *res, u32 vendor_type)

- int **wpa_scan_get_max_rate** (const struct wpa_scan_res *res)
- void **wpa_scan_results_free** (struct wpa_scan_results *res)
- void **wpa_scan_sort_results** (struct wpa_scan_results *res)

15.169.1 Detailed Description

WPA Supplicant - Helper functions for scan result processing.

Copyright

Copyright (c) 2007-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.170 src/eap_common/chap.c File Reference

CHAP-MD5 (RFC 1994). `#include "includes.h"`

`#include "common.h"`

`#include "md5.h"`

`#include "crypto.h"`

`#include "chap.h"`

Functions

- int **chap_md5** (u8 id, const u8 *secret, size_t secret_len, const u8 *challenge, size_t challenge_len, u8 *response)

15.170.1 Detailed Description

CHAP-MD5 (RFC 1994).

Copyright

Copyright (c) 2007-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.171 src/eap_common/chap.h File Reference

CHAP-MD5 (RFC 1994).

Defines

- #define **CHAP_MD5_LEN** 16

Functions

- int **chap_md5** (u8 id, const u8 ∗secret, size_t secret_len, const u8 ∗challenge, size_t challenge_len, u8 ∗response)

15.171.1 Detailed Description

CHAP-MD5 (RFC 1994).

Copyright

15.172 src/eap_common/eap_common.c File Reference

EAP common peer/server definitions. `#include "includes.h"`

`#include "common.h"`

`#include "eap_defs.h"`

`#include "eap_common.h"`

Functions

- const u8 * eap_hdr_validate (int vendor, EapType eap_type, const struct wpabuf *msg, size_t *plen)
  Validate EAP header.
- struct wpabuf * eap_msg_alloc (int vendor, EapType type, size_t payload_len, u8 code, u8 identifier)
  Allocate a buffer for an EAP message.
- void eap_update_len (struct wpabuf *msg)
  Update EAP header length.
- u8 eap_get_id (const struct wpabuf *msg)
  Get EAP Identifier from wpabuf.
- EapType eap_get_type (const struct wpabuf *msg)
  Get EAP Type from wpabuf.

15.172.1 Detailed Description

EAP common peer/server definitions.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.172.2 Function Documentation

15.172.2.1 u8 eap_get_id (const struct wpabuf * *msg*)

Get EAP Identifier from wpabuf.

Parameters:

msg Buffer starting with an EAP header

Returns:

The Identifier field from the EAP header

15.172.2.2 EapType eap_get_type (const struct wpabuf * *msg*)

Get EAP Type from wpabuf.

Parameters:

msg Buffer starting with an EAP header

Returns:

The EAP Type after the EAP header

15.172.2.3 const u8* eap_hdr_validate (int *vendor*, EapType *eap_type*, const struct wpabuf * *msg*, size_t * *plen*)

Validate EAP header.

Parameters:

vendor Expected EAP Vendor-Id (0 = IETF)

eap_type Expected EAP type number

msg EAP frame (starting with EAP header)

plen Pointer to variable to contain the returned payload length

Returns:

Pointer to EAP payload (after type field), or NULL on failure

This is a helper function for EAP method implementations. This is usually called in the beginning of struct eap_method::process() function to verify that the received EAP request packet has a valid header. This function is able to process both legacy and expanded EAP headers and in most cases, the caller can just use the returned payload pointer (into *plen) for processing the payload regardless of whether the packet used the expanded EAP header or not.

15.172.2.4 struct wpabuf* eap_msg_alloc (int *vendor*, EapType *type*, size_t *payload_len*, u8 *code*, u8 *identifier*) [read]

Allocate a buffer for an EAP message.

Parameters:

vendor Vendor-Id (0 = IETF)

type EAP type

payload_len Payload length in bytes (data after Type)

code Message Code (EAP_CODE_*)

identifier Identifier

Returns:

Pointer to the allocated message buffer or NULL on error

This function can be used to allocate a buffer for an EAP message and fill in the EAP header. This function is automatically using expanded EAP header if the selected Vendor-Id is not IETF. In other words, most EAP methods do not need to separately select which header type to use when using this function to allocate the message buffers. The returned buffer has room for payload_len bytes and has the EAP header and Type field already filled in.

15.172.2.5 void eap_update_len (struct wpabuf * *msg*)

Update EAP header length.

Parameters:

msg EAP message from eap_msg_alloc

This function updates the length field in the EAP header to match with the current length for the buffer. This allows eap_msg_alloc() to be used to allocate a larger buffer than the exact message length (e.g., if exact message length is not yet known).

15.173 src/eap_common/eap_common.h File Reference

EAP common peer/server definitions. `#include "wpabuf.h"`

Functions

- const u8 * eap_hdr_validate (int vendor, EapType eap_type, const struct wpabuf *msg, size_t *plen)

  Validate EAP header.

- struct wpabuf * eap_msg_alloc (int vendor, EapType type, size_t payload_len, u8 code, u8 identifier)

  Allocate a buffer for an EAP message.

- void eap_update_len (struct wpabuf *msg)

  Update EAP header length.

- u8 eap_get_id (const struct wpabuf *msg)

  Get EAP Identifier from wpabuf.

- EapType eap_get_type (const struct wpabuf *msg)

  Get EAP Type from wpabuf.

15.173.1 Detailed Description

EAP common peer/server definitions.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.173.2 Function Documentation

15.173.2.1 u8 eap_get_id (const struct wpabuf * *msg*)

Get EAP Identifier from wpabuf.

Parameters:

msg Buffer starting with an EAP header

Returns:

The Identifier field from the EAP header

15.173.2.2 EapType eap_get_type (const struct wpabuf * *msg*)

Get EAP Type from wpabuf.

Parameters:

msg Buffer starting with an EAP header

Returns:

The EAP Type after the EAP header

15.173.2.3 const u8* eap_hdr_validate (int *vendor*, EapType *eap_type*, const struct wpabuf * *msg*, size_t * *plen*)

Validate EAP header.

Parameters:

vendor Expected EAP Vendor-Id (0 = IETF)

eap_type Expected EAP type number

msg EAP frame (starting with EAP header)

plen Pointer to variable to contain the returned payload length

Returns:

Pointer to EAP payload (after type field), or NULL on failure

This is a helper function for EAP method implementations. This is usually called in the beginning of struct eap_method::process() function to verify that the received EAP request packet has a valid header. This function is able to process both legacy and expanded EAP headers and in most cases, the caller can just use the returned payload pointer (into *plen) for processing the payload regardless of whether the packet used the expanded EAP header or not.

15.173.2.4 struct wpabuf* eap_msg_alloc (int *vendor*, EapType *type*, size_t *payload_len*, u8 *code*, u8 *identifier*) [read]

Allocate a buffer for an EAP message.

Parameters:

vendor Vendor-Id (0 = IETF)

type EAP type

payload_len Payload length in bytes (data after Type)

code Message Code (EAP_CODE_*)

identifier Identifier

Returns:

Pointer to the allocated message buffer or NULL on error

This function can be used to allocate a buffer for an EAP message and fill in the EAP header. This function is automatically using expanded EAP header if the selected Vendor-Id is not IETF. In other words, most EAP methods do not need to separately select which header type to use when using this function to allocate the message buffers. The returned buffer has room for payload_len bytes and has the EAP header and Type field already filled in.

15.173.2.5 void eap_update_len (struct wpabuf * *msg*)

Update EAP header length.

Parameters:

msg EAP message from eap_msg_alloc

This function updates the length field in the EAP header to match with the current length for the buffer. This allows eap_msg_alloc() to be used to allocate a larger buffer than the exact message length (e.g., if exact message length is not yet known).

15.174 src/eap_common/eap_defs.h File Reference

EAP server/peer: Shared EAP definitions.

Data Structures

- struct eap_hdr

Defines

- #define **EAP_MSK_LEN** 64
- #define **EAP_EMSK_LEN** 64

Enumerations

- enum { **EAP_CODE_REQUEST** = 1, **EAP_CODE_RESPONSE** = 2, **EAP_CODE_SUCCESS** = 3, **EAP_CODE_FAILURE** = 4 }
- enum **EapType** {

  EAP_TYPE_NONE = 0, **EAP_TYPE_IDENTITY** = 1, **EAP_TYPE_NOTIFICATION** = 2, **EAP_TYPE_NAK** = 3,

  EAP_TYPE_MD5 = 4, **EAP_TYPE_OTP** = 5, **EAP_TYPE_GTC** = 6, **EAP_TYPE_TLS** = 13,

  EAP_TYPE_LEAP = 17, **EAP_TYPE_SIM** = 18, **EAP_TYPE_TTLS** = 21, **EAP_TYPE_AKA** = 23,

  EAP_TYPE_PEAP = 25, **EAP_TYPE_MSCHAPV2** = 26, **EAP_TYPE_TLV** = 33, **EAP_TYPE_TNC** = 38,

  EAP_TYPE_FAST = 43, **EAP_TYPE_PAX** = 46, **EAP_TYPE_PSK** = 47, **EAP_TYPE_SAKE** = 48,

  EAP_TYPE_IKEV2 = 49, **EAP_TYPE_AKA_PRIME** = 50, **EAP_TYPE_GPSK** = 51, **EAP_TYPE_EXPANDED** = 254 }
- enum { **EAP_VENDOR_IETF** = 0, **EAP_VENDOR_MICROSOFT** = 0x000137, **EAP_VENDOR_WFA** = 0x00372A }

Variables

- struct eap_hdr **STRUCT_PACKED**

15.174.1 Detailed Description

EAP server/peer: Shared EAP definitions.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.175 src/eap_common/eap_fast_common.c File Reference

EAP-FAST common helper functions (RFC 4851). `#include "includes.h"`

```
#include "common.h"
#include "sha1.h"
#include "tls.h"
#include "eap_defs.h"
#include "eap_tlv_common.h"
#include "eap_fast_common.h"
```

Defines

- #define **TLS_RANDOM_LEN** 32
- #define **TLS_MASTER_SECRET_LEN** 48

Functions

- void **eap_fast_put_tlv_hdr** (struct wpabuf *buf, u16 type, u16 len)
- void **eap_fast_put_tlv** (struct wpabuf *buf, u16 type, const void *data, u16 len)
- void **eap_fast_put_tlv_buf** (struct wpabuf *buf, u16 type, const struct wpabuf *data)
- struct wpabuf * **eap_fast_tlv_eap_payload** (struct wpabuf *buf)
- void **eap_fast_derive_master_secret** (const u8 *pac_key, const u8 *server_random, const u8 *client_random, u8 *master_secret)
- u8 * **eap_fast_derive_key** (void *ssl_ctx, struct tls_connection *conn, const char *label, size_t len)
- void **eap_fast_derive_eap_msk** (const u8 *simck, u8 *msk)
- void **eap_fast_derive_eap_emsk** (const u8 *simck, u8 *emsk)
- int **eap_fast_parse_tlv** (struct eap_fast_tlv_parse *tlv, int tlv_type, u8 *pos, int len)

15.175.1 Detailed Description

EAP-FAST common helper functions (RFC 4851).

Copyright

15.176 src/eap_common/eap_fast_common.h File Reference

EAP-FAST definitions (RFC 4851).

Data Structures

- struct pac_tlv_hdr
- struct eap_fast_key_block_provisioning
- struct eap_fast_tlv_parse

Defines

- #define **EAP_FAST_VERSION** 1
- #define **EAP_FAST_KEY_LEN** 64
- #define **EAP_FAST_SIMCK_LEN** 40
- #define **EAP_FAST_SKS_LEN** 40
- #define **EAP_FAST_CMK_LEN** 20
- #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_A_ID_INFO** 7
- #define **PAC_TYPE_PAC_ACKNOWLEDGEMENT** 8
- #define **PAC_TYPE_PAC_INFO** 9
- #define **PAC_TYPE_PAC_TYPE** 10
- #define **EAP_FAST_PAC_KEY_LEN** 32
- #define **PAC_TYPE_TUNNEL_PAC** 1
- #define **PAC_TYPE_USER_AUTHORIZATION** 3
- #define **PAC_TYPE_MACHINE_AUTHENTICATION** 2

Functions

- void **eap_fast_put_tlv_hdr** (struct wpabuf *buf, u16 type, u16 len)
- void **eap_fast_put_tlv** (struct wpabuf *buf, u16 type, const void *data, u16 len)
- void **eap_fast_put_tlv_buf** (struct wpabuf *buf, u16 type, const struct wpabuf *data)
- struct wpabuf * **eap_fast_tlv_eap_payload** (struct wpabuf *buf)
- void **eap_fast_derive_master_secret** (const u8 *pac_key, const u8 *server_random, const u8 *client_random, u8 *master_secret)
- u8 * **eap_fast_derive_key** (void *ssl_ctx, struct tls_connection *conn, const char *label, size_t len)
- void **eap_fast_derive_eap_msk** (const u8 *simck, u8 *msk)
- void **eap_fast_derive_eap_emsk** (const u8 *simck, u8 *emsk)
- int **eap_fast_parse_tlv** (struct eap_fast_tlv_parse *tlv, int tlv_type, u8 *pos, int len)

Variables

- struct pac_tlv_hdr **STRUCT_PACKED**

15.176.1 Detailed Description

EAP-FAST definitions (RFC 4851).

Copyright

15.177 src/eap_common/eap_gpsk_common.c File Reference

EAP server/peer: EAP-GPSK shared routines. `#include "includes.h"`

```
#include "common.h"
#include "eap_defs.h"
#include "aes_wrap.h"
#include "crypto.h"
#include "eap_gpsk_common.h"
```

Defines

- #define **EAP_GPSK_SK_LEN_AES** 16
- #define **EAP_GPSK_PK_LEN_AES** 16

Functions

- int eap_gpsk_supported_ciphersuite (int vendor, int specifier)

  Check whether ciphersuite is supported.

- int eap_gpsk_derive_keys (const u8 *psk, size_t psk_len, int vendor, int specifier, const u8 *rand_peer, const u8 *rand_server, const u8 *id_peer, size_t id_peer_len, const u8 *id_server, size_t id_server_len, u8 *msk, u8 *emsk, u8 *sk, size_t *sk_len, u8 *pk, size_t *pk_len)

  Derive EAP-GPSK keys.

- size_t eap_gpsk_mic_len (int vendor, int specifier)

  Get the length of the MIC.

- int eap_gpsk_compute_mic (const u8 *sk, size_t sk_len, int vendor, int specifier, const u8 *data, size_t len, u8 *mic)

  Compute EAP-GPSK MIC for an EAP packet.

15.177.1 Detailed Description

EAP server/peer: EAP-GPSK shared routines.

Copyright

15.177.2 Function Documentation

15.177.2.1 int eap_gpsk_compute_mic (const u8 * *sk*, size_t *sk_len*, int *vendor*, int *specifier*, const u8 * *data*, size_t *len*, u8 * *mic*)

Compute EAP-GPSK MIC for an EAP packet.

Parameters:

sk Session key SK from eap_gpsk_derive_keys()

sk_len SK length in bytes from eap_gpsk_derive_keys()

vendor CSuite/Vendor

specifier CSuite/Specifier

data Input data to MIC

len Input data length in bytes

mic Buffer for the computed MIC, eap_gpsk_mic_len(cipher) bytes

Returns:

0 on success, -1 on failure

15.177.2.2 int eap_gpsk_derive_keys (const u8 * *psk*, size_t *psk_len*, int *vendor*, int *specifier*, const u8 * *rand_peer*, const u8 * *rand_server*, const u8 * *id_peer*, size_t *id_peer_len*, const u8 * *id_server*, size_t *id_server_len*, u8 * *msk*, u8 * *emsk*, u8 * *sk*, size_t * *sk_len*, u8 * *pk*, size_t * *pk_len*)

Derive EAP-GPSK keys.

Parameters:

psk Pre-shared key

psk_len Length of psk in bytes

vendor CSuite/Vendor

specifier CSuite/Specifier

rand_peer 32-byte RAND_Peer

rand_server 32-byte RAND_Server

id_peer ID_Peer

id_peer_len Length of ID_Peer

id_server ID_Server

id_server_len Length of ID_Server

msk Buffer for 64-byte MSK

emsk Buffer for 64-byte EMSK

sk Buffer for SK (at least EAP_GPSK_MAX_SK_LEN bytes)

sk_len Buffer for returning length of SK

pk Buffer for PK (at least EAP_GPSK_MAX_PK_LEN bytes)

pk_len Buffer for returning length of PK

Returns:

0 on success, -1 on failure

15.177.2.3 size_t eap_gpsk_mic_len (int *vendor*, int *specifier*)

Get the length of the MIC.

Parameters:

vendor CSuite/Vendor

specifier CSuite/Specifier

Returns:

MIC length in bytes

15.177.2.4 int eap_gpsk_supported_ciphersuite (int *vendor*, int *specifier*)

Check whether ciphersuite is supported.

Parameters:

vendor CSuite/Vendor

specifier CSuite/Specifier

Returns:

1 if ciphersuite is support, or 0 if not

15.178 src/eap_common/eap_gpsk_common.h File Reference

EAP server/peer: EAP-GPSK shared routines.

Data Structures

- struct eap_gpsk_csuite

Defines

- #define **EAP_GPSK_OPCODE_GPSK_1** 1
- #define **EAP_GPSK_OPCODE_GPSK_2** 2
- #define **EAP_GPSK_OPCODE_GPSK_3** 3
- #define **EAP_GPSK_OPCODE_GPSK_4** 4
- #define **EAP_GPSK_OPCODE_FAIL** 5
- #define **EAP_GPSK_OPCODE_PROTECTED_FAIL** 6
- #define **EAP_GPSK_FAIL_PSK_NOT_FOUND** 0x00000001
- #define **EAP_GPSK_FAIL_AUTHENTICATION_FAILURE** 0x00000002
- #define **EAP_GPSK_FAIL_AUTHORIZATION_FAILURE** 0x00000003
- #define **EAP_GPSK_RAND_LEN** 32
- #define **EAP_GPSK_MAX_SK_LEN** 32
- #define **EAP_GPSK_MAX_PK_LEN** 32
- #define **EAP_GPSK_MAX_MIC_LEN** 32
- #define **EAP_GPSK_VENDOR_IETF** 0x00000000
- #define **EAP_GPSK_CIPHER_RESERVED** 0x000000
- #define **EAP_GPSK_CIPHER_AES** 0x000001
- #define **EAP_GPSK_CIPHER_SHA256** 0x000002

Functions

- int eap_gpsk_supported_ciphersuite (int vendor, int specifier)

  Check whether ciphersuite is supported.

- int eap_gpsk_derive_keys (const u8 *psk, size_t psk_len, int vendor, int specifier, const u8 *rand_client, const u8 *rand_server, const u8 *id_client, size_t id_client_len, const u8 *id_server, size_t id_server_len, u8 *msk, u8 *emsk, u8 *sk, size_t *sk_len, u8 *pk, size_t *pk_len)

  Derive EAP-GPSK keys.

- size_t eap_gpsk_mic_len (int vendor, int specifier)

  Get the length of the MIC.

- int eap_gpsk_compute_mic (const u8 *sk, size_t sk_len, int vendor, int specifier, const u8 *data, size_t len, u8 *mic)

  Compute EAP-GPSK MIC for an EAP packet.

Variables

- struct eap_gpsk_csuite **STRUCT_PACKED**

15.178.1 Detailed Description

EAP server/peer: EAP-GPSK shared routines.

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.178.2 Function Documentation

15.178.2.1 int eap_gpsk_compute_mic (const u8 * *sk*, size_t *sk_len*, int *vendor*, int *specifier*, const u8 * *data*, size_t *len*, u8 * *mic*)

Compute EAP-GPSK MIC for an EAP packet.

Parameters:

sk Session key SK from eap_gpsk_derive_keys()

sk_len SK length in bytes from eap_gpsk_derive_keys()

vendor CSuite/Vendor

specifier CSuite/Specifier

data Input data to MIC

len Input data length in bytes

mic Buffer for the computed MIC, eap_gpsk_mic_len(cipher) bytes

Returns:

0 on success, -1 on failure

15.178.2.2 int eap_gpsk_derive_keys (const u8 * *psk*, size_t *psk_len*, int *vendor*, int *specifier*, const u8 * *rand_peer*, const u8 * *rand_server*, const u8 * *id_peer*, size_t *id_peer_len*, const u8 * *id_server*, size_t *id_server_len*, u8 * *msk*, u8 * *emsk*, u8 * *sk*, size_t * *sk_len*, u8 * *pk*, size_t * *pk_len*)

Derive EAP-GPSK keys.

Parameters:

psk Pre-shared key

psk_len Length of psk in bytes

vendor CSuite/Vendor

specifier CSuite/Specifier

rand_peer 32-byte RAND_Peer

rand_server 32-byte RAND_Server

id_peer ID_Peer

id_peer_len Length of ID_Peer

id_server ID_Server

id_server_len Length of ID_Server

msk Buffer for 64-byte MSK

emsk Buffer for 64-byte EMSK

sk Buffer for SK (at least EAP_GPSK_MAX_SK_LEN bytes)

sk_len Buffer for returning length of SK

pk Buffer for PK (at least EAP_GPSK_MAX_PK_LEN bytes)

pk_len Buffer for returning length of PK

Returns:

0 on success, -1 on failure

15.178.2.3 size_t eap_gpsk_mic_len (int *vendor*, int *specifier*)

Get the length of the MIC.

Parameters:

vendor CSuite/Vendor

specifier CSuite/Specifier

Returns:

MIC length in bytes

15.178.2.4 int eap_gpsk_supported_ciphersuite (int *vendor*, int *specifier*)

Check whether ciphersuite is supported.

Parameters:

vendor CSuite/Vendor

specifier CSuite/Specifier

Returns:

1 if ciphersuite is support, or 0 if not

15.179 src/eap_common/eap_ikev2_common.c File Reference

EAP-IKEv2 common routines. `#include "includes.h"`

`#include "common.h"`

`#include "eap_defs.h"`

`#include "eap_common.h"`

`#include "ikev2_common.h"`

`#include "eap_ikev2_common.h"`

Functions

- int **eap_ikev2_derive_keymat** (int prf, struct ikev2_keys *keys, const u8 *i_nonce, size_t i_nonce_-len, const u8 *r_nonce, size_t r_nonce_len, u8 *keymat)
- struct wpabuf * **eap_ikev2_build_frag_ack** (u8 id, u8 code)
- int **eap_ikev2_validate_icv** (int integ_alg, struct ikev2_keys *keys, int initiator, const struct wpabuf *msg, const u8 *pos, const u8 *end)

15.179.1 Detailed Description

EAP-IKEv2 common routines.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.180 src/eap_common/eap_ikev2_common.h File Reference

EAP-IKEv2 definitions.

Defines

- #define **IKEV2_FLAGS_LENGTH_INCLUDED** 0x80
- #define **IKEV2_FLAGS_MORE_FRAGMENTS** 0x40
- #define **IKEV2_FLAGS_ICV_INCLUDED** 0x20
- #define **IKEV2_FRAGMENT_SIZE** 1400

Functions

- int **eap_ikev2_derive_keymat** (int prf, struct ikev2_keys *keys, const u8 *i_nonce, size_t i_nonce_-len, const u8 *r_nonce, size_t r_nonce_len, u8 *keymat)
- struct wpabuf * **eap_ikev2_build_frag_ack** (u8 id, u8 code)
- int **eap_ikev2_validate_icv** (int integ_alg, struct ikev2_keys *keys, int initiator, const struct wpabuf *msg, const u8 *pos, const u8 *end)

15.180.1 Detailed Description

EAP-IKEv2 definitions.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.181 src/eap_common/eap_pax_common.c File Reference

EAP server/peer: EAP-PAX shared routines. `#include "includes.h"`

`#include "common.h"`

`#include "sha1.h"`

`#include "eap_pax_common.h"`

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.

15.181.1 Detailed Description

EAP server/peer: EAP-PAX shared routines.

Copyright

Copyright (c) 2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.181.2 Function Documentation

15.181.2.1 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

15.181.2.2 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

RFC 4746, Section 2.6: PAX-KDF-W(X, Y, Z)

15.181.2.3 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.

15.182 src/eap_common/eap_pax_common.h File Reference

EAP server/peer: EAP-PAX shared routines.

Data Structures

- struct eap_pax_hdr

Defines

- #define **EAP_PAX_FLAGS_MF** 0x01
- #define **EAP_PAX_FLAGS_CE** 0x02
- #define **EAP_PAX_FLAGS_AI** 0x04
- #define **EAP_PAX_MAC_HMAC_SHA1_128** 0x01
- #define **EAP_PAX_HMAC_SHA256_128** 0x02
- #define **EAP_PAX_DH_GROUP_NONE** 0x00
- #define **EAP_PAX_DH_GROUP_2048_MODP** 0x01
- #define **EAP_PAX_DH_GROUP_3072_MODP** 0x02
- #define **EAP_PAX_DH_GROUP_NIST_ECC_P_256** 0x03
- #define **EAP_PAX_PUBLIC_KEY_NONE** 0x00
- #define **EAP_PAX_PUBLIC_KEY_RSAES_OAEP** 0x01
- #define **EAP_PAX_PUBLIC_KEY_RSA_PKCS1_V1_5** 0x02
- #define **EAP_PAX_PUBLIC_KEY_EL_GAMAL_NIST_ECC** 0x03
- #define **EAP_PAX_ADE_VENDOR_SPECIFIC** 0x01
- #define **EAP_PAX_ADE_CLIENT_CHANNEL_BINDING** 0x02
- #define **EAP_PAX_ADE_SERVER_CHANNEL_BINDING** 0x03
- #define **EAP_PAX_RAND_LEN** 32
- #define **EAP_PAX_MAC_LEN** 16
- #define **EAP_PAX_ICV_LEN** 16
- #define **EAP_PAX_AK_LEN** 16
- #define **EAP_PAX_MK_LEN** 16
- #define **EAP_PAX_CK_LEN** 16
- #define **EAP_PAX_ICK_LEN** 16

Enumerations

- enum {

  EAP_PAX_OP_STD_1 = 0x01, **EAP_PAX_OP_STD_2** = 0x02, **EAP_PAX_OP_STD_3** = 0x03, **EAP_PAX_OP_SEC_1** = 0x11,

  EAP_PAX_OP_SEC_2 = 0x12, **EAP_PAX_OP_SEC_3** = 0x13, **EAP_PAX_OP_SEC_4** = 0x14, **EAP_PAX_OP_SEC_5** = 0x15,

  EAP_PAX_OP_ACK = 0x21 }

Functions

- int eap_pax_kdf (u8 mac_id, const u8 *key, size_t key_len, const char *identifier, const u8 *entropy, size_t entropy_len, size_t output_len, u8 *output)
  PAX Key Derivation Function.
- int eap_pax_mac (u8 mac_id, const u8 *key, size_t key_len, const u8 *data1, size_t data1_len, const u8 *data2, size_t data2_len, const u8 *data3, size_t data3_len, u8 *mac)
  EAP-PAX MAC.
- int eap_pax_initial_key_derivation (u8 mac_id, const u8 *ak, const u8 *e, u8 *mk, u8 *ck, u8 *ick)
  EAP-PAX initial key derivation.

Variables

- struct eap_pax_hdr **STRUCT_PACKED**

15.182.1 Detailed Description

EAP server/peer: EAP-PAX shared routines.

Copyright

Copyright (c) 2005-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.182.2 Function Documentation

15.182.2.1 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

15.182.2.2 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

RFC 4746, Section 2.6: PAX-KDF-W(X, Y, Z)

15.182.2.3 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.

15.183 src/eap_common/eap_peap_common.c File Reference

EAP-PEAP common routines. `#include "includes.h"`

`#include "common.h"`

`#include "sha1.h"`

`#include "eap_peap_common.h"`

Functions

- void **peap_prfplus** (int version, const u8 *key, size_t key_len, const char *label, const u8 *seed, size_t seed_len, u8 *buf, size_t buf_len)

15.183.1 Detailed Description

EAP-PEAP common routines.

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

15.184 src/eap_common/eap_peap_common.h File Reference

EAP-PEAP common routines.

Functions

- void **peap_prfplus** (int version, const u8 *key, size_t key_len, const char *label, const u8 *seed, size_t seed_len, u8 *buf, size_t buf_len)

15.184.1 Detailed Description

EAP-PEAP common routines.

Copyright

15.185 src/eap_common/eap_psk_common.c File Reference

EAP server/peer: EAP-PSK shared routines. `#include "includes.h"`

`#include "common.h"`

`#include "aes_wrap.h"`

`#include "eap_defs.h"`

`#include "eap_psk_common.h"`

Defines

- #define **aes_block_size** 16

Functions

- int **eap_psk_key_setup** (const u8 *psk, u8 *ak, u8 *kdk)
- int **eap_psk_derive_keys** (const u8 *kdk, const u8 *rand_p, u8 *tek, u8 *msk, u8 *emsk)

15.185.1 Detailed Description

EAP server/peer: EAP-PSK shared routines.

Copyright

Copyright (c) 2004-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.186 src/eap_common/eap_psk_common.h File Reference

EAP server/peer: EAP-PSK shared routines.

Data Structures

- struct eap_psk_hdr_1
- struct eap_psk_hdr_2
- struct eap_psk_hdr_3
- struct eap_psk_hdr_4

Defines

- #define **EAP_PSK_RAND_LEN** 16
- #define **EAP_PSK_MAC_LEN** 16
- #define **EAP_PSK_TEK_LEN** 16
- #define **EAP_PSK_PSK_LEN** 16
- #define **EAP_PSK_AK_LEN** 16
- #define **EAP_PSK_KDK_LEN** 16
- #define **EAP_PSK_R_FLAG_CONT** 1
- #define **EAP_PSK_R_FLAG_DONE_SUCCESS** 2
- #define **EAP_PSK_R_FLAG_DONE_FAILURE** 3
- #define **EAP_PSK_E_FLAG** 0x20
- #define **EAP_PSK_FLAGS_GET_T**(flags) (((flags) & 0xc0) >> 6)
- #define **EAP_PSK_FLAGS_SET_T**(t) ((u8) (t) << 6)

Functions

- int __must_check **eap_psk_key_setup** (const u8 *psk, u8 *ak, u8 *kdk)
- int __must_check **eap_psk_derive_keys** (const u8 *kdk, const u8 *rand_p, u8 *tek, u8 *msk, u8 *emsk)

Variables

- struct eap_psk_hdr_1 **STRUCT_PACKED**

15.186.1 Detailed Description

EAP server/peer: EAP-PSK shared routines.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.187 src/eap_common/eap_sake_common.c File Reference

EAP server/peer: EAP-SAKE shared routines. `#include "includes.h"`

```
#include "common.h"
#include "sha1.h"
#include "wpabuf.h"
#include "eap_defs.h"
#include "eap_sake_common.h"
```

Functions

- int eap_sake_parse_attributes (const u8 *buf, size_t len, struct eap_sake_parse_attr *attr)
  Parse EAP-SAKE attributes.
- void eap_sake_derive_keys (const u8 *root_secret_a, const u8 *root_secret_b, const u8 *rand_s, const u8 *rand_p, u8 *tek, u8 *msk, u8 *emsk)
  Derive EAP-SAKE keys.
- int eap_sake_compute_mic (const u8 *tek_auth, const u8 *rand_s, const u8 *rand_p, const u8 *serverid, size_t serverid_len, const u8 *peerid, size_t peerid_len, int peer, const u8 *eap, size_t eap_len, const u8 *mic_pos, u8 *mic)
  Compute EAP-SAKE MIC for an EAP packet.
- void **eap_sake_add_attr** (struct wpabuf *buf, u8 type, const u8 *data, size_t len)

15.187.1 Detailed Description

EAP server/peer: EAP-SAKE shared routines.

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.187.2 Function Documentation

15.187.2.1 int eap_sake_compute_mic (const u8 * *tek_auth*, const u8 * *rand_s*, const u8 * *rand_p*, const u8 * *serverid*, size_t *serverid_len*, const u8 * *peerid*, size_t *peerid_len*, int *peer*, const u8 * *eap*, size_t *eap_len*, const u8 * *mic_pos*, u8 * *mic*)

Compute EAP-SAKE MIC for an EAP packet.

Parameters:

tek_auth 16-byte TEK-Auth

rand_s 16-byte RAND_S

rand_p 16-byte RAND_P

serverid SERVERID

serverid_len SERVERID length

peerid PEERID

peerid_len PEERID length

peer MIC calculation for 0 = Server, 1 = Peer message

eap EAP packet

eap_len EAP packet length

mic_pos MIC position in the EAP packet (must be [eap .. eap + eap_len])

mic Buffer for the computed 16-byte MIC

15.187.2.2 void eap_sake_derive_keys (const u8 * *root_secret_a*, const u8 * *root_secret_b*, const u8 * *rand_s*, const u8 * *rand_p*, u8 * *tek*, u8 * *msk*, u8 * *emsk*)

Derive EAP-SAKE keys.

Parameters:

root_secret_a 16-byte Root-Secret-A

root_secret_b 16-byte Root-Secret-B

rand_s 16-byte RAND_S

rand_p 16-byte RAND_P

tek Buffer for Temporary EAK Keys (TEK-Auth[16] | TEK-Cipher[16])

msk Buffer for 64-byte MSK

emsk Buffer for 64-byte EMSK

This function derives EAP-SAKE keys as defined in RFC 4763, section 3.2.6.

15.187.2.3 int eap_sake_parse_attributes (const u8 * *buf*, size_t *len*, struct eap_sake_parse_attr * *attr*)

Parse EAP-SAKE attributes.

Parameters:

buf Packet payload (starting with the first attribute)

len Payload length

attr Structure to be filled with found attributes

Returns:

0 on success or -1 on failure

15.188 src/eap_common/eap_sake_common.h File Reference

EAP server/peer: EAP-SAKE shared routines.

Data Structures

- struct eap_sake_hdr
- struct eap_sake_parse_attr

Defines

- #define **EAP_SAKE_VERSION** 2
- #define **EAP_SAKE_SUBTYPE_CHALLENGE** 1
- #define **EAP_SAKE_SUBTYPE_CONFIRM** 2
- #define **EAP_SAKE_SUBTYPE_AUTH_REJECT** 3
- #define **EAP_SAKE_SUBTYPE_IDENTITY** 4
- #define **EAP_SAKE_AT_RAND_S** 1
- #define **EAP_SAKE_AT_RAND_P** 2
- #define **EAP_SAKE_AT_MIC_S** 3
- #define **EAP_SAKE_AT_MIC_P** 4
- #define **EAP_SAKE_AT_SERVERID** 5
- #define **EAP_SAKE_AT_PEERID** 6
- #define **EAP_SAKE_AT_SPI_S** 7
- #define **EAP_SAKE_AT_SPI_P** 8
- #define **EAP_SAKE_AT_ANY_ID_REQ** 9
- #define **EAP_SAKE_AT_PERM_ID_REQ** 10
- #define **EAP_SAKE_AT_ENCR_DATA** 128
- #define **EAP_SAKE_AT_IV** 129
- #define **EAP_SAKE_AT_PADDING** 130
- #define **EAP_SAKE_AT_NEXT_TMPID** 131
- #define **EAP_SAKE_AT_MSK_LIFE** 132
- #define **EAP_SAKE_RAND_LEN** 16
- #define **EAP_SAKE_MIC_LEN** 16
- #define **EAP_SAKE_ROOT_SECRET_LEN** 16
- #define **EAP_SAKE_SMS_LEN** 16
- #define **EAP_SAKE_TEK_AUTH_LEN** 16
- #define **EAP_SAKE_TEK_CIPHER_LEN** 16
- #define **EAP_SAKE_TEK_LEN** (EAP_SAKE_TEK_AUTH_LEN + EAP_SAKE_TEK_-CIPHER_LEN)

Functions

- int eap_sake_parse_attributes (const u8 *buf, size_t len, struct eap_sake_parse_attr *attr)

  Parse EAP-SAKE attributes.

- void eap_sake_derive_keys (const u8 *root_secret_a, const u8 *root_secret_b, const u8 *rand_s, const u8 *rand_p, u8 *tek, u8 *msk, u8 *emsk)

  Derive EAP-SAKE keys.

- int eap_sake_compute_mic (const u8 *tek_auth, const u8 *rand_s, const u8 *rand_p, const u8 *serverid, size_t serverid_len, const u8 *peerid, size_t peerid_len, int peer, const u8 *eap, size_t eap_len, const u8 *mic_pos, u8 *mic)

  Compute EAP-SAKE MIC for an EAP packet.

- void **eap_sake_add_attr** (struct wpabuf *buf, u8 type, const u8 *data, size_t len)

Variables

- struct eap_sake_hdr **STRUCT_PACKED**

15.188.1 Detailed Description

EAP server/peer: EAP-SAKE shared routines.

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.188.2 Function Documentation

15.188.2.1 int eap_sake_compute_mic (const u8 * *tek_auth*, const u8 * *rand_s*, const u8 * *rand_p*, const u8 * *serverid*, size_t *serverid_len*, const u8 * *peerid*, size_t *peerid_len*, int *peer*, const u8 * *eap*, size_t *eap_len*, const u8 * *mic_pos*, u8 * *mic*)

Compute EAP-SAKE MIC for an EAP packet.

Parameters:

tek_auth 16-byte TEK-Auth

rand_s 16-byte RAND_S

rand_p 16-byte RAND_P

serverid SERVERID

serverid_len SERVERID length

peerid PEERID

peerid_len PEERID length

peer MIC calculation for 0 = Server, 1 = Peer message

eap EAP packet

eap_len EAP packet length

mic_pos MIC position in the EAP packet (must be [eap .. eap + eap_len])

mic Buffer for the computed 16-byte MIC

15.188.2.2 void eap_sake_derive_keys (const u8 * *root_secret_a*, const u8 * *root_secret_b*, const u8 * *rand_s*, const u8 * *rand_p*, u8 * *tek*, u8 * *msk*, u8 * *emsk*)

Derive EAP-SAKE keys.

Parameters:

root_secret_a 16-byte Root-Secret-A

root_secret_b 16-byte Root-Secret-B

rand_s 16-byte RAND_S

rand_p 16-byte RAND_P

tek Buffer for Temporary EAK Keys (TEK-Auth[16] | TEK-Cipher[16])

msk Buffer for 64-byte MSK

emsk Buffer for 64-byte EMSK

This function derives EAP-SAKE keys as defined in RFC 4763, section 3.2.6.

15.188.2.3 int eap_sake_parse_attributes (const u8 * *buf*, size_t *len*, struct eap_sake_parse_attr * *attr*)

Parse EAP-SAKE attributes.

Parameters:

buf Packet payload (starting with the first attribute)

len Payload length

attr Structure to be filled with found attributes

Returns:

0 on success or -1 on failure

15.189 src/eap_common/eap_sim_common.c File Reference

EAP peer/server: EAP-SIM/AKA/AKA' shared routines. `#include "includes.h"`

```
#include "common.h"
#include "eap_common/eap_defs.h"
#include "sha1.h"
#include "sha256.h"
#include "crypto.h"
#include "aes_wrap.h"
#include "wpabuf.h"
#include "eap_common/eap_sim_common.h"
```

Data Structures

- struct eap_sim_msg

Defines

- #define **EAP_SIM_INIT_LEN** 128

Functions

- void **eap_sim_derive_mk** (const u8 *identity, size_t identity_len, const u8 *nonce_mt, u16 selected_version, const u8 *ver_list, size_t ver_list_len, int num_chal, const u8 *kc, u8 *mk)
- void **eap_aka_derive_mk** (const u8 *identity, size_t identity_len, const u8 *ik, const u8 *ck, u8 *mk)
- int **eap_sim_derive_keys** (const u8 *mk, u8 *k_encr, u8 *k_aut, u8 *msk, u8 *emsk)
- int **eap_sim_derive_keys_reauth** (u16 _counter, const u8 *identity, size_t identity_len, const u8 *nonce_s, const u8 *mk, u8 *msk, u8 *emsk)
- int **eap_sim_verify_mac** (const u8 *k_aut, const struct wpabuf *req, const u8 *mac, const u8 *extra, size_t extra_len)
- void **eap_sim_add_mac** (const u8 *k_aut, const 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)
- struct eap_sim_msg * **eap_sim_msg_init** (int code, int id, int type, int subtype)
- struct wpabuf * **eap_sim_msg_finish** (struct eap_sim_msg *msg, 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)

15.189.1 Detailed Description

EAP peer/server: EAP-SIM/AKA/AKA' shared routines.

Copyright

15.190 src/eap_common/eap_sim_common.h File Reference

EAP peer/server: EAP-SIM/AKA/AKA' shared routines.

Data Structures

- struct eap_sim_attrs

Defines

- #define **EAP_SIM_NONCE_S_LEN** 16
- #define **EAP_SIM_NONCE_MT_LEN** 16
- #define **EAP_SIM_MAC_LEN** 16
- #define **EAP_SIM_MK_LEN** 20
- #define **EAP_SIM_K_AUT_LEN** 16
- #define **EAP_SIM_K_ENCR_LEN** 16
- #define **EAP_SIM_KEYING_DATA_LEN** 64
- #define **EAP_SIM_IV_LEN** 16
- #define **EAP_SIM_KC_LEN** 8
- #define **EAP_SIM_SRES_LEN** 4
- #define **GSM_RAND_LEN** 16
- #define **EAP_SIM_VERSION** 1
- #define **EAP_SIM_SUBTYPE_START** 10
- #define **EAP_SIM_SUBTYPE_CHALLENGE** 11
- #define **EAP_SIM_SUBTYPE_NOTIFICATION** 12
- #define **EAP_SIM_SUBTYPE_REAUTHENTICATION** 13
- #define **EAP_SIM_SUBTYPE_CLIENT_ERROR** 14
- #define **EAP_SIM_UNABLE_TO_PROCESS_PACKET** 0
- #define **EAP_SIM_UNSUPPORTED_VERSION** 1
- #define **EAP_SIM_INSUFFICIENT_NUM_OF_CHAL** 2
- #define **EAP_SIM_RAND_NOT_FRESH** 3
- #define **EAP_SIM_MAX_FAST_REAUTHS** 1000
- #define **EAP_SIM_MAX_CHAL** 3
- #define **EAP_AKA_SUBTYPE_CHALLENGE** 1
- #define **EAP_AKA_SUBTYPE_AUTHENTICATION_REJECT** 2
- #define **EAP_AKA_SUBTYPE_SYNCHRONIZATION_FAILURE** 4
- #define **EAP_AKA_SUBTYPE_IDENTITY** 5
- #define **EAP_AKA_SUBTYPE_NOTIFICATION** 12
- #define **EAP_AKA_SUBTYPE_REAUTHENTICATION** 13
- #define **EAP_AKA_SUBTYPE_CLIENT_ERROR** 14
- #define **EAP_AKA_UNABLE_TO_PROCESS_PACKET** 0
- #define **EAP_AKA_RAND_LEN** 16
- #define **EAP_AKA_AUTN_LEN** 16
- #define **EAP_AKA_AUTS_LEN** 14
- #define **EAP_AKA_RES_MAX_LEN** 16
- #define **EAP_AKA_IK_LEN** 16
- #define **EAP_AKA_CK_LEN** 16
- #define **EAP_AKA_MAX_FAST_REAUTHS** 1000
- #define **EAP_AKA_MIN_RES_LEN** 4

- #define **EAP_AKA_MAX_RES_LEN** 16
- #define **EAP_AKA_CHECKCODE_LEN** 20
- #define **EAP_AKA_PRIME_K_AUT_LEN** 32
- #define **EAP_AKA_PRIME_CHECKCODE_LEN** 32
- #define **EAP_AKA_PRIME_K_RE_LEN** 32
- #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_KDF_INPUT** 23
- #define **EAP_SIM_AT_KDF** 24
- #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_AT_BIDDING** 136
- #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
- #define **EAP_AKA_PRIME_KDF** 1
- #define **EAP_AKA_BIDDING_FLAG_D** 0x8000
- #define **EAP_AKA_PRIME_KDF_MAX** 10

Enumerations

- enum **eap_sim_id_req** { **NO_ID_REQ**, **ANY_ID**, **FULLAUTH_ID**, **PERMANENT_ID** }

Functions

- void **eap_sim_derive_mk** (const u8 *identity, size_t identity_len, const u8 *nonce_mt, u16 selected_version, const u8 *ver_list, size_t ver_list_len, int num_chal, const u8 *kc, u8 *mk)
- void **eap_aka_derive_mk** (const u8 *identity, size_t identity_len, const u8 *ik, const u8 *ck, u8 *mk)
- int **eap_sim_derive_keys** (const u8 *mk, u8 *k_encr, u8 *k_aut, u8 *msk, u8 *emsk)
- int **eap_sim_derive_keys_reauth** (u16 _counter, const u8 *identity, size_t identity_len, const u8 *nonce_s, const u8 *mk, u8 *msk, u8 *emsk)
- int **eap_sim_verify_mac** (const u8 *k_aut, const struct wpabuf *req, const u8 *mac, const u8 *extra, size_t extra_len)
- void **eap_sim_add_mac** (const u8 *k_aut, const 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)
- struct eap_sim_msg * **eap_sim_msg_init** (int code, int id, int type, int subtype)
- struct wpabuf * **eap_sim_msg_finish** (struct eap_sim_msg *msg, 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)

15.190.1 Detailed Description

EAP peer/server: EAP-SIM/AKA/AKA' shared routines.

Copyright

15.191 src/eap_common/eap_tlv_common.h File Reference

EAP-TLV definitions (draft-josefsson-pppext-eap-tls-eap-10.txt).

Data Structures

- struct eap_tlv_hdr
- struct eap_tlv_nak_tlv
- struct eap_tlv_result_tlv
- struct eap_tlv_intermediate_result_tlv
- struct eap_tlv_crypto_binding_tlv
- struct eap_tlv_pac_ack_tlv
- struct eap_tlv_request_action_tlv
- struct eap_tlv_pac_type_tlv

Defines

- #define **EAP_TLV_RESULT_TLV** 3
- #define **EAP_TLV_NAK_TLV** 4
- #define **EAP_TLV_ERROR_CODE_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_CALLING_STATION_ID_TLV** 13
- #define **EAP_TLV_CALLED_STATION_ID_TLV** 14
- #define **EAP_TLV_NAS_PORT_TYPE_TLV** 15
- #define **EAP_TLV_SERVER_IDENTIFIER_TLV** 16
- #define **EAP_TLV_IDENTITY_TYPE_TLV** 17
- #define **EAP_TLV_SERVER_TRUSTED_ROOT_TLV** 18
- #define **EAP_TLV_REQUEST_ACTION_TLV** 19
- #define **EAP_TLV_PKCS7_TLV** 20
- #define **EAP_TLV_RESULT_SUCCESS** 1
- #define **EAP_TLV_RESULT_FAILURE** 2
- #define **EAP_TLV_TYPE_MANDATORY** 0x8000
- #define **EAP_TLV_TYPE_MASK** 0x3fff
- #define **EAP_TLV_CRYPTO_BINDING_SUBTYPE_REQUEST** 0
- #define **EAP_TLV_CRYPTO_BINDING_SUBTYPE_RESPONSE** 1
- #define **EAP_TLV_ACTION_PROCESS_TLV** 1
- #define **EAP_TLV_ACTION_NEGOTIATE_EAP** 2

Variables

- struct eap_tlv_hdr **STRUCT_PACKED**

15.191.1 Detailed Description

EAP-TLV definitions (draft-josefsson-pppext-eap-tls-eap-10.txt).

Copyright

15.192 src/eap_common/eap_ttls.h File Reference

EAP server/peer: EAP-TTLS (RFC 5281).

Data Structures

- struct ttls_avp
- struct ttls_avp_vendor

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

15.192.1 Detailed Description

EAP server/peer: EAP-TTLS (RFC 5281).

Copyright

15.192.2 Define Documentation

15.192.2.1 #define AVP_PAD(start, pos)

Value:

```
do { \
        int __pad; \
        __pad = (4 - (((pos) - (start)) & 3)) & 3; \
        os_memset((pos), 0, __pad); \
        pos += __pad; \
} while (0)
```

15.193 src/eap_common/eap_wsc_common.c File Reference

EAP-WSC common routines for Wi-Fi Protected Setup. `#include "includes.h"`

```
#include "common.h"
#include "eap_defs.h"
#include "eap_common.h"
#include "wps/wps.h"
#include "eap_wsc_common.h"
```

Functions

- struct wpabuf * **eap_wsc_build_frag_ack** (u8 id, u8 code)

15.193.1 Detailed Description

EAP-WSC common routines for Wi-Fi Protected Setup.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.194 src/eap_common/eap_wsc_common.h File Reference

EAP-WSC definitions for Wi-Fi Protected Setup.

Defines

- #define **EAP_VENDOR_TYPE_WSC** 1
- #define **WSC_FLAGS_MF** 0x01
- #define **WSC_FLAGS_LF** 0x02
- #define **WSC_ID_REGISTRAR** "WFA-SimpleConfig-Registrar-1-0"
- #define **WSC_ID_REGISTRAR_LEN** 30
- #define **WSC_ID_ENROLLEE** "WFA-SimpleConfig-Enrollee-1-0"
- #define **WSC_ID_ENROLLEE_LEN** 29
- #define **WSC_FRAGMENT_SIZE** 1400

Functions

- struct wpabuf * **eap_wsc_build_frag_ack** (u8 id, u8 code)

15.194.1 Detailed Description

EAP-WSC definitions for Wi-Fi Protected Setup.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.195 src/eap_common/ikev2_common.c File Reference

IKEv2 common routines for initiator and responder. `#include "includes.h"`

```
#include "common.h"
#include "sha1.h"
#include "md5.h"
#include "crypto.h"
#include "ikev2_common.h"
```

Defines

- #define **NUM_INTEG_ALGS** (sizeof(ikev2_integ_algs) / sizeof(ikev2_integ_algs[0]))
- #define **NUM_PRF_ALGS** (sizeof(ikev2_prf_algs) / sizeof(ikev2_prf_algs[0]))
- #define **NUM_ENCR_ALGS** (sizeof(ikev2_encr_algs) / sizeof(ikev2_encr_algs[0]))

Functions

- struct ikev2_integ_alg * **ikev2_get_integ** (int id)
- int **ikev2_integ_hash** (int alg, const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *hash)
- struct ikev2_prf_alg * **ikev2_get_prf** (int id)
- int **ikev2_prf_hash** (int alg, const u8 *key, size_t key_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *hash)
- int **ikev2_prf_plus** (int alg, const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *out, size_t out_len)
- struct ikev2_encr_alg * **ikev2_get_encr** (int id)
- int **ikev2_encr_encrypt** (int alg, const u8 *key, size_t key_len, const u8 *iv, const u8 *plain, u8 *crypt, size_t len)
- int **ikev2_encr_decrypt** (int alg, const u8 *key, size_t key_len, const u8 *iv, const u8 *crypt, u8 *plain, size_t len)
- int **ikev2_parse_payloads** (struct ikev2_payloads *payloads, u8 next_payload, const u8 *pos, const u8 *end)
- int **ikev2_derive_auth_data** (int prf_alg, const struct wpabuf *sign_msg, const u8 *ID, size_t ID_-len, u8 ID_type, struct ikev2_keys *keys, int initiator, const u8 *shared_secret, size_t shared_secret_-len, const u8 *nonce, size_t nonce_len, const u8 *key_pad, size_t key_pad_len, u8 *auth_data)
- u8 * **ikev2_decrypt_payload** (int encr_id, int integ_id, struct ikev2_keys *keys, int initiator, const struct ikev2_hdr *hdr, const u8 *encrypted, size_t encrypted_len, size_t *res_len)
- void **ikev2_update_hdr** (struct wpabuf *msg)
- int **ikev2_build_encrypted** (int encr_id, int integ_id, struct ikev2_keys *keys, int initiator, struct wpabuf *msg, struct wpabuf *plain, u8 next_payload)
- int **ikev2_keys_set** (struct ikev2_keys *keys)
- void **ikev2_free_keys** (struct ikev2_keys *keys)
- int **ikev2_derive_sk_keys** (const struct ikev2_prf_alg *prf, const struct ikev2_integ_alg *integ, const struct ikev2_encr_alg *encr, const u8 *skeyseed, const u8 *data, size_t data_len, struct ikev2_keys *keys)

15.195.1 Detailed Description

IKEv2 common routines for initiator and responder.

Copyright

15.196 src/eap_common/ikev2_common.h File Reference

IKEv2 definitions.

Data Structures

- struct ikev2_hdr
- struct ikev2_payload_hdr
- struct ikev2_proposal
- struct ikev2_transform
- struct ikev2_keys
- struct ikev2_integ_alg
- struct ikev2_prf_alg
- struct ikev2_encr_alg
- struct ikev2_payloads

Defines

- #define **IKEV2_NONCE_MIN_LEN** 16
- #define **IKEV2_NONCE_MAX_LEN** 256
- #define **IKEV2_SPI_LEN** 8
- #define **IKEV2_MjVer** 2
- #define **IKEV2_MnVer** 0
- #define **IKEV2_VERSION** (((IKEV2_MjVer) << 4) | (IKEV2_MnVer))
- #define **IKEV2_HDR_INITIATOR** 0x08
- #define **IKEV2_HDR_VERSION** 0x10
- #define **IKEV2_HDR_RESPONSE** 0x20
- #define **IKEV2_PAYLOAD_FLAGS_CRITICAL** 0x01
- #define **IKEV2_MAX_HASH_LEN** 20

Enumerations

- enum { **IKE_SA_INIT** = 34, **IKE_SA_AUTH** = 35, **CREATE_CHILD_SA** = 36, **INFORMATION** = 37 }
- enum {

  IKEV2_PAYLOAD_NO_NEXT_PAYLOAD = 0, **IKEV2_PAYLOAD_SA** = 33, **IKEV2_PAYLOAD_KEY_EXCHANGE** = 34, **IKEV2_PAYLOAD_IDi** = 35,

  IKEV2_PAYLOAD_IDr = 36, **IKEV2_PAYLOAD_CERTIFICATE** = 37, **IKEV2_PAYLOAD_CERT_REQ** = 38, **IKEV2_PAYLOAD_AUTHENTICATION** = 39,

  IKEV2_PAYLOAD_NONCE = 40, **IKEV2_PAYLOAD_NOTIFICATION** = 41, **IKEV2_PAYLOAD_VENDOD_ID** = 43, **IKEV2_PAYLOAD_ENCRYPTED** = 46,

  IKEV2_PAYLOAD_NEXT_FAST_ID = 121 }
- enum { **IKEV2_PROTOCOL_RESERVED** = 0, **IKEV2_PROTOCOL_IKE** = 1, **IKEV2_PROTOCOL_AH** = 2, **IKEV2_PROTOCOL_ESP** = 3 }
- enum {

  IKEV2_TRANSFORM_ENCR = 1, **IKEV2_TRANSFORM_PRF** = 2, **IKEV2_TRANSFORM_INTEG** = 3, **IKEV2_TRANSFORM_DH** = 4,

  IKEV2_TRANSFORM_ESN = 5 }

- enum {

  ENCR_DES_IV64 = 1, **ENCR_DES** = 2, **ENCR_3DES** = 3, **ENCR_RC5** = 4,

  ENCR_IDEA = 5, **ENCR_CAST** = 6, **ENCR_BLOWFISH** = 7, **ENCR_3IDEA** = 8,

  ENCR_DES_IV32 = 9, **ENCR_NULL** = 11, **ENCR_AES_CBC** = 12, **ENCR_AES_CTR** = 13 }
- enum { **PRF_HMAC_MD5** = 1, **PRF_HMAC_SHA1** = 2, **PRF_HMAC_TIGER** = 3, **PRF_-AES128_XCBC** = 4 }
- enum {

  AUTH_HMAC_MD5_96 = 1, **AUTH_HMAC_SHA1_96** = 2, **AUTH_DES_MAC** = 3, **AUTH_-KPDK_MD5** = 4,

  AUTH_AES_XCBC_96 = 5 }
- enum {

  DH_GROUP1_768BIT_MODP = 1, **DH_GROUP2_1024BIT_MODP** = 2, **DH_GROUP5_-1536BIT_MODP** = 5, **DH_GROUP5_2048BIT_MODP** = 14,

  DH_GROUP5_3072BIT_MODP = 15, **DH_GROUP5_4096BIT_MODP** = 16, **DH_GROUP5_-6144BIT_MODP** = 17, **DH_GROUP5_8192BIT_MODP** = 18 }
- enum {

  ID_IPV4_ADDR = 1, **ID_FQDN** = 2, **ID_RFC822_ADDR** = 3, **ID_IPV6_ADDR** = 5,

  ID_DER_ASN1_DN = 9, **ID_DER_ASN1_GN** = 10, **ID_KEY_ID** = 11 }
- enum {

  CERT_ENCODING_PKCS7_X509 = 1, **CERT_ENCODING_PGP_CERT** = 2, **CERT_-ENCODING_DNS_SIGNED_KEY** = 3, **CERT_ENCODING_X509_CERT_SIGN** = 4,

  CERT_ENCODING_KERBEROS_TOKEN = 6, **CERT_ENCODING_CRL** = 7, **CERT_-ENCODING_ARL** = 8, **CERT_ENCODING_SPKI_CERT** = 9,

  CERT_ENCODING_X509_CERT_ATTR = 10, **CERT_ENCODING_RAW_RSA_KEY** = 11, **CERT_ENCODING_HASH_AND_URL_X509_CERT** = 12, **CERT_ENCODING_HASH_-AND_URL_X509_BUNDLE** = 13 }
- enum { **AUTH_RSA_SIGN** = 1, **AUTH_SHARED_KEY_MIC** = 2, **AUTH_DSS_SIGN** = 3 }
- enum {

  UNSUPPORTED_CRITICAL_PAYLOAD = 1, **INVALID_IKE_SPI** = 4, **INVALID_MAJOR_-VERSION** = 5, **INVALID_SYNTAX** = 7,

  INVALID_MESSAGE_ID = 9, **INVALID_SPI** = 11, **NO_PROPOSAL_CHOSEN** = 14, **INVALID_KE_PAYLOAD** = 17,

  AUTHENTICATION_FAILED = 24, **SINGLE_PAIR_REQUIRED** = 34, **NO_-ADDITIONAL_SAS** = 35, **INTERNAL_ADDRESS_FAILURE** = 36,

  FAILED_CP_REQUIRED = 37, **TS_UNACCEPTABLE** = 38, **INVALID_SELECTORS** = 39 }

Functions

- int **ikev2_keys_set** (struct ikev2_keys *keys)
- void **ikev2_free_keys** (struct ikev2_keys *keys)
- struct ikev2_integ_alg * **ikev2_get_integ** (int id)
- int **ikev2_integ_hash** (int alg, const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *hash)
- struct ikev2_prf_alg * **ikev2_get_prf** (int id)
- int **ikev2_prf_hash** (int alg, const u8 *key, size_t key_len, size_t num_elem, const u8 *addr[ ], const size_t *len, u8 *hash)

- int **ikev2_prf_plus** (int alg, const u8 *key, size_t key_len, const u8 *data, size_t data_len, u8 *out, size_t out_len)
- struct ikev2_encr_alg * **ikev2_get_encr** (int id)
- int **ikev2_encr_encrypt** (int alg, const u8 *key, size_t key_len, const u8 *iv, const u8 *plain, u8 *crypt, size_t len)
- int **ikev2_encr_decrypt** (int alg, const u8 *key, size_t key_len, const u8 *iv, const u8 *crypt, u8 *plain, size_t len)
- int **ikev2_derive_auth_data** (int prf_alg, const struct wpabuf *sign_msg, const u8 *ID, size_t ID_len, u8 ID_type, struct ikev2_keys *keys, int initiator, const u8 *shared_secret, size_t shared_secret_len, const u8 *nonce, size_t nonce_len, const u8 *key_pad, size_t key_pad_len, u8 *auth_data)
- int **ikev2_parse_payloads** (struct ikev2_payloads *payloads, u8 next_payload, const u8 *pos, const u8 *end)
- u8 * **ikev2_decrypt_payload** (int encr_id, int integ_id, struct ikev2_keys *keys, int initiator, const struct ikev2_hdr *hdr, const u8 *encrypted, size_t encrypted_len, size_t *res_len)
- void **ikev2_update_hdr** (struct wpabuf *msg)
- int **ikev2_build_encrypted** (int encr_id, int integ_id, struct ikev2_keys *keys, int initiator, struct wpabuf *msg, struct wpabuf *plain, u8 next_payload)
- int **ikev2_derive_sk_keys** (const struct ikev2_prf_alg *prf, const struct ikev2_integ_alg *integ, const struct ikev2_encr_alg *encr, const u8 *skeyseed, const u8 *data, size_t data_len, struct ikev2_keys *keys)

Variables

- struct ikev2_hdr **STRUCT_PACKED**

15.196.1 Detailed Description

IKEv2 definitions.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.197 src/eap_peer/eap.c File Reference

EAP peer state machines (RFC 4137). `#include "includes.h"`

```
#include "common.h"
#include "eap_i.h"
#include "eap_config.h"
#include "tls.h"
#include "crypto.h"
#include "pcsc_funcs.h"
#include "wpa_ctrl.h"
#include "state_machine.h"
#include "eap_common/eap_wsc_common.h"
```

Defines

- #define **STATE_MACHINE_DATA** struct eap_sm
- #define **STATE_MACHINE_DEBUG_PREFIX** "EAP"
- #define **EAP_MAX_AUTH_ROUNDS** 50

Enumerations

- enum **eap_ctrl_req_type** {
  TYPE_IDENTITY, **TYPE_PASSWORD**, **TYPE_OTP**, **TYPE_PIN**,
  TYPE_NEW_PASSWORD, **TYPE_PASSPHRASE** }

Functions

- int eap_allowed_method (struct eap_sm *sm, int vendor, u32 method)

  Check whether EAP method is allowed.

- **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)
- struct wpabuf * eap_sm_buildIdentity (struct eap_sm *sm, int id, int encrypted)

Build EAP-Identity/Response for the current network.

- struct eap_sm * eap_peer_sm_init (void *eapol_ctx, struct eapol_callbacks *eapol_cb, void *msg_ctx, struct eap_config *conf)

  Allocate and initialize EAP peer state machine.

- void eap_peer_sm_deinit (struct eap_sm *sm)

  Deinitialize and free an EAP peer state machine.

- int eap_peer_sm_step (struct eap_sm *sm)

  Step EAP peer state machine.

- void eap_sm_abort (struct eap_sm *sm)

  Abort EAP authentication.

- int eap_sm_get_status (struct eap_sm *sm, char *buf, size_t buflen, int verbose)

  Get EAP state machine status.

- void eap_sm_request_identity (struct eap_sm *sm)

  Request identity from user (ctrl_iface).

- void eap_sm_request_password (struct eap_sm *sm)

  Request password from user (ctrl_iface).

- void eap_sm_request_new_password (struct eap_sm *sm)

  Request new password from user (ctrl_iface).

- void eap_sm_request_pin (struct eap_sm *sm)

  Request SIM or smart card PIN from user (ctrl_iface).

- void eap_sm_request_otp (struct eap_sm *sm, const char *msg, size_t msg_len)

  Request one time password from user (ctrl_iface).

- void eap_sm_request_passphrase (struct eap_sm *sm)

  Request passphrase from user (ctrl_iface).

- void eap_sm_notify_ctrl_attached (struct eap_sm *sm)

  Notification of attached monitor.

- u32 eap_get_phase2_type (const char *name, int *vendor)

  Get EAP type for the given EAP phase 2 method name.

- struct eap_method_type * eap_get_phase2_types (struct eap_peer_config *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.

- struct eap_peer_config * eap_get_config (struct eap_sm *sm)
  Get current network configuration.
- const u8 * eap_get_config_identity (struct eap_sm *sm, size_t *len)
  Get identity from the network configuration.
- const u8 * eap_get_config_password (struct eap_sm *sm, size_t *len)
  Get password from the network configuration.
- const u8 * eap_get_config_password2 (struct eap_sm *sm, size_t *len, int *hash)
  Get password from the network configuration.
- const u8 * eap_get_config_new_password (struct eap_sm *sm, size_t *len)
  Get new password from network configuration.
- const u8 * eap_get_config_otp (struct eap_sm *sm, size_t *len)
  Get one-time password from the network configuration.
- void eap_clear_config_otp (struct eap_sm *sm)
  Clear used one-time password.
- const char * eap_get_config_phase1 (struct eap_sm *sm)
  Get phase1 data from the network configuration.
- const char * eap_get_config_phase2 (struct eap_sm *sm)
  Get phase2 data from the 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.
- struct wpabuf * eap_get_eapRespData (struct eap_sm *sm)
  Get EAP response data.
- void eap_register_scard_ctx (struct eap_sm *sm, void *ctx)
  Notification of smart card context.
- void eap_set_config_blob (struct eap_sm *sm, struct wpa_config_blob *blob)
  Set or add a named configuration blob.
- 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.

- void eap_notify_pending (struct eap_sm *sm)

  Notify that EAP method is ready to re-process a request.

- void eap_invalidate_cached_session (struct eap_sm *sm)

  Mark cached session data invalid.

- int **eap_is_wps_pbc_enrollee** (struct eap_peer_config *conf)
- int **eap_is_wps_pin_enrollee** (struct eap_peer_config *conf)

15.197.1 Detailed Description

EAP peer state machines (RFC 4137).

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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).

15.197.2 Function Documentation

15.197.2.1 int eap_allowed_method (struct eap_sm * *sm*, int *vendor*, u32 *method*)

Check whether EAP method is allowed.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

vendor Vendor-Id for expanded types or 0 = IETF for legacy types

method EAP type

Returns:

1 = allowed EAP method, 0 = not allowed

15.197.2.2 void eap_clear_config_otp (struct eap_sm * *sm*)

Clear used one-time password.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

This function clears a used one-time password (OTP) from the current network configuration. This should be called when the OTP has been used and is not needed anymore.

15.197.2.3 struct eap_peer_config* eap_get_config (struct eap_sm * *sm*) [read]

Get current network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Returns:

Pointer to the current network configuration or NULL if not found

EAP peer methods should avoid using this function if they can use other access functions, like eap_get_config_identity() and eap_get_config_password(), that do not require direct access to struct eap_peer_config.

15.197.2.4 struct wpa_config_blob* eap_get_config_blob (struct eap_sm * *sm*, const char * *name*) [read]

Get a named configuration blob.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

name Name of the blob

Returns:

Pointer to blob data or NULL if not found

15.197.2.5 const u8* eap_get_config_identity (struct eap_sm * *sm*, size_t * *len*)

Get identity from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the identity

Returns:

Pointer to the identity or NULL if not found

15.197.2.6 const u8* eap_get_config_new_password (struct eap_sm * *sm*, size_t * *len*)

Get new password from network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the new password

Returns:

Pointer to the new password or NULL if not found

15.197.2.7 const u8* eap_get_config_otp (struct eap_sm * *sm*, size_t * *len*)

Get one-time password from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the one-time password

Returns:

Pointer to the one-time password or NULL if not found

15.197.2.8 const u8* eap_get_config_password (struct eap_sm * *sm*, size_t * *len*)

Get password from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the password

Returns:

Pointer to the password or NULL if not found

15.197.2.9 const u8* eap_get_config_password2 (struct eap_sm * *sm*, size_t * *len*, int * *hash*)

Get password from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the password

hash Buffer for returning whether the password is stored as a NtPasswordHash instead of plaintext password; can be NULL if this information is not needed

Returns:

Pointer to the password or NULL if not found

15.197.2.10 const char* eap_get_config_phase1 (struct eap_sm * *sm*)

Get phase1 data from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Returns:

Pointer to the phase1 data or NULL if not found

15.197.2.11 const char* eap_get_config_phase2 (struct eap_sm * *sm*)

Get phase2 data from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Returns:

Pointer to the phase1 data or NULL if not found

15.197.2.12 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_peer_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.

15.197.2.13 struct wpabuf* eap_get_eapRespData (struct eap_sm * *sm*) [read]

Get EAP response data.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

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.

15.197.2.14 u32 eap_get_phase2_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP phase 2 method name.

Parameters:

name EAP method name, e.g., MD5

vendor Buffer for returning EAP Vendor-Id

Returns:

EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers that are allowed for Phase 2, i.e., for tunneled authentication. Phase 2 is used, e.g., with EAP-PEAP, EAP-TTLS, and EAP-FAST.

15.197.2.15 struct eap_method_type* eap_get_phase2_types (struct eap_peer_config * *config*, size_t * *count*) [read]

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.

15.197.2.16 void eap_invalidate_cached_session (struct eap_sm * *sm*)

Mark cached session data invalid.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

15.197.2.17 int eap_key_available (struct eap_sm * *sm*)

Get key availability (eapKeyAvailable variable).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Returns:

1 if EAP keying material is available, 0 if not

15.197.2.18 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_peer_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.

15.197.2.19 void eap_notify_pending (struct eap_sm * *sm*)

Notify that EAP method is ready to re-process a request.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

An EAP method can perform a pending operation (e.g., to get a response from an external process). Once the response is available, this function can be used to request EAPOL state machine to retry delivering the previously received (and still unanswered) EAP request to EAP state machine.

15.197.2.20 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_peer_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).

15.197.2.21 void eap_peer_sm_deinit (struct eap_sm * *sm*)

Deinitialize and free an EAP peer state machine.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

This function deinitializes EAP state machine and frees all allocated resources.

15.197.2.22 struct eap_sm* eap_peer_sm_init (void * *eapol_ctx*, struct eapol_callbacks * *eapol_cb*, void * *msg_ctx*, struct eap_config * *conf*) [read]

Allocate and initialize EAP peer 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_peer_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.

15.197.2.23 int eap_peer_sm_step (struct eap_sm * *sm*)

Step EAP peer state machine.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

15.197.2.24 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_peer_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.

15.197.2.25 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_peer_sm_init()

blob New value for the blob

Adds a new configuration blob or replaces the current value of an existing blob.

15.197.2.26 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_peer_sm_init()

enabled 1 = Fast reauthentication is enabled, 0 = Disabled

15.197.2.27 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_peer_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).

15.197.2.28 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_peer_sm_init()

workaround 1 = Enable EAP workarounds, 0 = Disable EAP workarounds

15.197.2.29 void eap_sm_abort (struct eap_sm * *sm*)

Abort EAP authentication.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Release system resources that have been allocated for the authentication session without fully deinitializing the EAP state machine.

15.197.2.30 struct wpabuf* eap_sm_buildIdentity (struct eap_sm * *sm*, int *id*, int *encrypted*) [read]

Build EAP-Identity/Response for the current network.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

id EAP identifier for the packet

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.

15.197.2.31 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_peer_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.

15.197.2.32 void eap_sm_notify_ctrl_attached (struct eap_sm * *sm*)

Notification of attached monitor.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

15.197.2.33 void eap_sm_request_identity (struct eap_sm * *sm*)

Request identity from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request identity information for the current network. This is normally called when the identity is not included in the network configuration. The request will be sent to monitor programs through the control interface.

15.197.2.34 void eap_sm_request_new_password (struct eap_sm * *sm*)

Request new password from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request new password information for the current network. This is normally called when the EAP method indicates that the current password has expired and password change is required. The request will be sent to monitor programs through the control interface.

15.197.2.35 void eap_sm_request_otp (struct eap_sm * *sm*, const char * *msg*, size_t *msg_len*)

Request one time password from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

msg Message to be displayed to the user when asking for OTP

msg_len Length of the user displayable message

EAP methods can call this function to request open time password (OTP) for the current network. The request will be sent to monitor programs through the control interface.

15.197.2.36 void eap_sm_request_passphrase (struct eap_sm * *sm*)

Request passphrase from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request passphrase for a private key for the current network. This is normally called when the passphrase is not included in the network configuration. The request will be sent to monitor programs through the control interface.

15.197.2.37 void eap_sm_request_password (struct eap_sm * *sm*)

Request password from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request password information for the current network. This is normally called when the password is not included in the network configuration. The request will be sent to monitor programs through the control interface.

15.197.2.38 void eap_sm_request_pin (struct eap_sm * *sm*)

Request SIM or smart card PIN from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request SIM or smart card PIN information for the current network. This is normally called when the PIN is not included in the network configuration. The request will be sent to monitor programs through the control interface.

15.198 src/eap_server/eap.c File Reference

hostapd / EAP Full Authenticator state machine (RFC 4137) `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

`#include "state_machine.h"`

Defines

- #define **STATE_MACHINE_DATA** struct eap_sm
- #define **STATE_MACHINE_DEBUG_PREFIX** "EAP"
- #define **EAP_MAX_AUTH_ROUNDS** 50
- #define **EAP_COPY**(dst, src) eap_copy_data((dst), (dst ## Len), (src), (src ## Len))

Functions

- int eap_user_get (struct eap_sm *sm, const u8 *identity, size_t identity_len, int phase2)

  Fetch user information from the database.

- **SM_STATE** (EAP, DISABLED)
- **SM_STATE** (EAP, INITIALIZE)
- **SM_STATE** (EAP, PICK_UP_METHOD)
- **SM_STATE** (EAP, IDLE)
- **SM_STATE** (EAP, RETRANSMIT)
- **SM_STATE** (EAP, RECEIVED)
- **SM_STATE** (EAP, DISCARD)
- **SM_STATE** (EAP, SEND_REQUEST)
- **SM_STATE** (EAP, INTEGRITY_CHECK)
- **SM_STATE** (EAP, METHOD_REQUEST)
- **SM_STATE** (EAP, METHOD_RESPONSE)
- **SM_STATE** (EAP, PROPOSE_METHOD)
- **SM_STATE** (EAP, NAK)
- **SM_STATE** (EAP, SELECT_ACTION)
- **SM_STATE** (EAP, TIMEOUT_FAILURE)
- **SM_STATE** (EAP, FAILURE)
- **SM_STATE** (EAP, SUCCESS)
- **SM_STATE** (EAP, INITIALIZE_PASSTHROUGH)
- **SM_STATE** (EAP, IDLE2)
- **SM_STATE** (EAP, RETRANSMIT2)
- **SM_STATE** (EAP, RECEIVED2)
- **SM_STATE** (EAP, DISCARD2)
- **SM_STATE** (EAP, SEND_REQUEST2)
- **SM_STATE** (EAP, AAA_REQUEST)
- **SM_STATE** (EAP, AAA_RESPONSE)
- **SM_STATE** (EAP, AAA_IDLE)
- **SM_STATE** (EAP, TIMEOUT_FAILURE2)
- **SM_STATE** (EAP, FAILURE2)
- **SM_STATE** (EAP, SUCCESS2)

- **SM_STEP** (EAP)
- void eap_sm_process_nak (struct eap_sm *sm, const u8 *nak_list, size_t len)

  Process EAP-Response/Nak.

- int eap_server_sm_step (struct eap_sm *sm)

  Step EAP server state machine.

- struct eap_sm * eap_server_sm_init (void *eapol_ctx, struct eapol_callbacks *eapol_cb, struct eap_config *conf)

  Allocate and initialize EAP server state machine.

- void eap_server_sm_deinit (struct eap_sm *sm)

  Deinitialize and free an EAP server state machine.

- void eap_sm_notify_cached (struct eap_sm *sm)

  Notify EAP state machine of cached PMK.

- void eap_sm_pending_cb (struct eap_sm *sm)

  EAP state machine callback for a pending EAP request.

- int eap_sm_method_pending (struct eap_sm *sm)

  Query whether EAP method is waiting for pending data.

- const u8 * eap_get_identity (struct eap_sm *sm, size_t *len)

  Get the user identity (from EAP-Response/Identity).

- struct eap_eapol_interface * eap_get_interface (struct eap_sm *sm)

  Get pointer to EAP-EAPOL interface data.

15.198.1 Detailed Description

hostapd / EAP Full Authenticator state machine (RFC 4137)

Copyright

This state machine is based on the full authenticator state machine defined in RFC 4137. However, to support backend authentication in RADIUS authentication server functionality, parts of backend authenticator (also from RFC 4137) are mixed in. This functionality is enabled by setting backend_auth configuration variable to TRUE.

15.198.2 Function Documentation

15.198.2.1 const u8* eap_get_identity (struct eap_sm * *sm*, size_t * *len*)

Get the user identity (from EAP-Response/Identity).

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

len Buffer for returning identity length

Returns:

Pointer to the user identity or NULL if not available

15.198.2.2 struct eap_eapol_interface* eap_get_interface (struct eap_sm * *sm*) [read]

Get pointer to EAP-EAPOL interface data.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

Returns:

Pointer to the EAP-EAPOL interface data

15.198.2.3 void eap_server_sm_deinit (struct eap_sm * *sm*)

Deinitialize and free an EAP server state machine.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

This function deinitializes EAP state machine and frees all allocated resources.

15.198.2.4 struct eap_sm* eap_server_sm_init (void * *eapol_ctx*, struct eapol_callbacks * *eapol_cb*, struct eap_config * *conf*) [read]

Allocate and initialize EAP server state machine.

Parameters:

eapol_ctx Context data to be used with eapol_cb calls

eapol_cb Pointer to EAPOL callback functions

conf EAP configuration

Returns:

Pointer to the allocated EAP state machine or NULL on failure

This function allocates and initializes an EAP state machine.

15.198.2.5 int eap_server_sm_step (struct eap_sm * *sm*)

Step EAP server state machine.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_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.

15.198.2.6 int eap_sm_method_pending (struct eap_sm * *sm*)

Query whether EAP method is waiting for pending data.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

Returns:

1 if method is waiting for pending data or 0 if not

15.198.2.7 void eap_sm_notify_cached (struct eap_sm * *sm*)

Notify EAP state machine of cached PMK.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

This function is called when PMKSA caching is used to skip EAP authentication.

15.198.2.8 void eap_sm_pending_cb (struct eap_sm * *sm*)

EAP state machine callback for a pending EAP request.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

This function is called when data for a pending EAP-Request is received.

15.198.2.9 void eap_sm_process_nak (struct eap_sm * *sm*, const u8 * *nak_list*, size_t *len*)

Process EAP-Response/Nak.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

nak_list Nak list (allowed methods) from the supplicant

len Length of nak_list in bytes

This function is called when EAP-Response/Nak is received from the supplicant. This can happen for both phase 1 and phase 2 authentications.

15.198.2.10 int eap_user_get (struct eap_sm * *sm*, const u8 * *identity*, size_t *identity_len*, int *phase2*)

Fetch user information from the database.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

identity Identity (User-Name) of the user

identity_len Length of identity in bytes

phase2 0 = EAP phase1 user, 1 = EAP phase2 (tunneled) user

Returns:

0 on success, or -1 on failure

This function is used to fetch user information for EAP. The user will be selected based on the specified identity. sm->user and sm->user_eap_method_index are updated for the new user when a matching user is found. sm->user can be used to get user information (e.g., password).

15.199 src/eap_peer/eap.h File Reference

EAP peer state machine functions (RFC 4137). `#include "defs.h"`

`#include "eap_common/eap_defs.h"`

`#include "eap_peer/eap_methods.h"`

Data Structures

- struct eap_method_type
- struct eapol_callbacks

  Callback functions from EAP to lower layer.

- struct eap_config

  Configuration for EAP state machine.

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

- struct eap_sm * eap_peer_sm_init (void *eapol_ctx, struct eapol_callbacks *eapol_cb, void *msg_ctx, struct eap_config *conf)

  Allocate and initialize EAP peer state machine.

- void eap_peer_sm_deinit (struct eap_sm *sm)

  Deinitialize and free an EAP peer state machine.

- int eap_peer_sm_step (struct eap_sm *sm)

  Step EAP peer 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.

- struct wpabuf * eap_sm_buildIdentity (struct eap_sm *sm, int id, int encrypted)

  Build EAP-Identity/Response for the current network.

- void eap_sm_request_identity (struct eap_sm *sm)

  Request identity from user (ctrl_iface).

- void eap_sm_request_password (struct eap_sm *sm)
  Request password from user (ctrl_iface).
- void eap_sm_request_new_password (struct eap_sm *sm)
  Request new password from user (ctrl_iface).
- void eap_sm_request_pin (struct eap_sm *sm)
  Request SIM or smart card PIN from user (ctrl_iface).
- void eap_sm_request_otp (struct eap_sm *sm, const char *msg, size_t msg_len)
  Request one time password from user (ctrl_iface).
- void eap_sm_request_passphrase (struct eap_sm *sm)
  Request passphrase from user (ctrl_iface).
- void eap_sm_notify_ctrl_attached (struct eap_sm *sm)
  Notification of attached monitor.
- u32 eap_get_phase2_type (const char *name, int *vendor)
  Get EAP type for the given EAP phase 2 method name.
- struct eap_method_type * eap_get_phase2_types (struct eap_peer_config *config, size_t *count)
  Get list of allowed EAP phase 2 types.
- void eap_set_fast_reauth (struct eap_sm *sm, int enabled)
  Update fast_reauth setting.
- void eap_set_workaround (struct eap_sm *sm, unsigned int workaround)
  Update EAP workarounds setting.
- void eap_set_force_disabled (struct eap_sm *sm, int disabled)
  Set force_disabled flag.
- int eap_key_available (struct eap_sm *sm)
  Get key availability (eapKeyAvailable variable).
- void eap_notify_success (struct eap_sm *sm)
  Notify EAP state machine about external success trigger.
- void eap_notify_lower_layer_success (struct eap_sm *sm)
  Notification of lower layer success.
- const u8 * eap_get_eapKeyData (struct eap_sm *sm, size_t *len)
  Get master session key (MSK) from EAP state machine.
- struct wpabuf * eap_get_eapRespData (struct eap_sm *sm)
  Get EAP response data.
- void eap_register_scard_ctx (struct eap_sm *sm, void *ctx)

Notification of smart card context.

- void eap_invalidate_cached_session (struct eap_sm *sm)

  Mark cached session data invalid.

- int **eap_is_wps_pbc_enrollee** (struct eap_peer_config *conf)
- int **eap_is_wps_pin_enrollee** (struct eap_peer_config *conf)

15.199.1 Detailed Description

EAP peer state machine functions (RFC 4137).

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.199.2 Enumeration Type Documentation

15.199.2.1 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.

Enumerator:

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.

15.199.2.2 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.

Enumerator:

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.

15.199.3 Function Documentation

15.199.3.1 const u8* eap_get_eapKeyData (struct eap_sm * *sm*, size_t * *len*)

Get master session key (MSK) from EAP state machine.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

15.199.3.2 struct wpabuf* eap_get_eapRespData (struct eap_sm * *sm*) [read]

Get EAP response data.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

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.

15.199.3.3 u32 eap_get_phase2_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP phase 2 method name.

Parameters:

name EAP method name, e.g., MD5

vendor Buffer for returning EAP Vendor-Id

Returns:

EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers that are allowed for Phase 2, i.e., for tunneled authentication. Phase 2 is used, e.g., with EAP-PEAP, EAP-TTLS, and EAP-FAST.

15.199.3.4 struct eap_method_type* eap_get_phase2_types (struct eap_peer_config * *config*, size_t * *count*) [read]

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.

15.199.3.5 void eap_invalidate_cached_session (struct eap_sm * *sm*)

Mark cached session data invalid.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

15.199.3.6 int eap_key_available (struct eap_sm * *sm*)

Get key availability (eapKeyAvailable variable).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Returns:

1 if EAP keying material is available, 0 if not

15.199.3.7 void eap_notify_lower_layer_success (struct eap_sm * *sm*)

Notification of lower layer success.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

15.199.3.8 void eap_notify_success (struct eap_sm * *sm*)

Notify EAP state machine about external success trigger.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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).

15.199.3.9 void eap_peer_sm_deinit (struct eap_sm * *sm*)

Deinitialize and free an EAP peer state machine.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

This function deinitializes EAP state machine and frees all allocated resources.

15.199.3.10 struct eap_sm* eap_peer_sm_init (void * *eapol_ctx*, struct eapol_callbacks * *eapol_cb*, void * *msg_ctx*, struct eap_config * *conf*) [read]

Allocate and initialize EAP peer 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_peer_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.

15.199.3.11 int eap_peer_sm_step (struct eap_sm * *sm*)

Step EAP peer state machine.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

15.199.3.12 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_peer_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.

15.199.3.13 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_peer_sm_init()

enabled 1 = Fast reauthentication is enabled, 0 = Disabled

15.199.3.14 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_peer_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).

15.199.3.15 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_peer_sm_init()

workaround 1 = Enable EAP workarounds, 0 = Disable EAP workarounds

15.199.3.16 void eap_sm_abort (struct eap_sm * *sm*)

Abort EAP authentication.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Release system resources that have been allocated for the authentication session without fully deinitializing the EAP state machine.

15.199.3.17 struct wpabuf* eap_sm_buildIdentity (struct eap_sm * *sm*, int *id*, int *encrypted*) [read]

Build EAP-Identity/Response for the current network.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

id EAP identifier for the packet

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.

15.199.3.18 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_peer_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.

15.199.3.19 void eap_sm_notify_ctrl_attached (struct eap_sm * *sm*)

Notification of attached monitor.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_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.

15.199.3.20 void eap_sm_request_identity (struct eap_sm * *sm*)

Request identity from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request identity information for the current network. This is normally called when the identity is not included in the network configuration. The request will be sent to monitor programs through the control interface.

15.199.3.21 void eap_sm_request_new_password (struct eap_sm * *sm*)

Request new password from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request new password information for the current network. This is normally called when the EAP method indicates that the current password has expired and password change is required. The request will be sent to monitor programs through the control interface.

15.199.3.22 void eap_sm_request_otp (struct eap_sm * *sm*, const char * *msg*, size_t *msg_len*)

Request one time password from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

msg Message to be displayed to the user when asking for OTP

msg_len Length of the user displayable message

EAP methods can call this function to request open time password (OTP) for the current network. The request will be sent to monitor programs through the control interface.

15.199.3.23 void eap_sm_request_passphrase (struct eap_sm * *sm*)

Request passphrase from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request passphrase for a private key for the current network. This is normally called when the passphrase is not included in the network configuration. The request will be sent to monitor programs through the control interface.

15.199.3.24 void eap_sm_request_password (struct eap_sm * *sm*)

Request password from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request password information for the current network. This is normally called when the password is not included in the network configuration. The request will be sent to monitor programs through the control interface.

15.199.3.25 void eap_sm_request_pin (struct eap_sm * *sm*)

Request SIM or smart card PIN from user (ctrl_iface).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

EAP methods can call this function to request SIM or smart card PIN information for the current network. This is normally called when the PIN is not included in the network configuration. The request will be sent to monitor programs through the control interface.

15.200 src/eap_server/eap.h File Reference

hostapd / EAP Full Authenticator state machine (RFC 4137) `#include "defs.h"`

`#include "eap_common/eap_defs.h"`

`#include "eap_server/eap_methods.h"`

`#include "wpabuf.h"`

Data Structures

- struct eap_user
- struct eap_eapol_interface
- struct eapol_callbacks

  Callback functions from EAP to lower layer.

- struct eap_config

  Configuration for EAP state machine.

Defines

- #define **EAP_MAX_METHODS** 8
- #define **EAP_TTLS_AUTH_PAP** 1
- #define **EAP_TTLS_AUTH_CHAP** 2
- #define **EAP_TTLS_AUTH_MSCHAP** 4
- #define **EAP_TTLS_AUTH_MSCHAPV2** 8

Functions

- struct eap_sm * eap_server_sm_init (void *eapol_ctx, struct eapol_callbacks *eapol_cb, struct eap_config *eap_conf)

  Allocate and initialize EAP server state machine.

- void eap_server_sm_deinit (struct eap_sm *sm)

  Deinitialize and free an EAP server state machine.

- int eap_server_sm_step (struct eap_sm *sm)

  Step EAP server state machine.

- void eap_sm_notify_cached (struct eap_sm *sm)

  Notify EAP state machine of cached PMK.

- void eap_sm_pending_cb (struct eap_sm *sm)

  EAP state machine callback for a pending EAP request.

- int eap_sm_method_pending (struct eap_sm *sm)

  Query whether EAP method is waiting for pending data.

- const u8 * eap_get_identity (struct eap_sm *sm, size_t *len)

  Get the user identity (from EAP-Response/Identity).

- struct eap_eapol_interface * eap_get_interface (struct eap_sm *sm)

  Get pointer to EAP-EAPOL interface data.

15.200.1 Detailed Description

hostapd / EAP Full Authenticator state machine (RFC 4137)

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.200.2 Function Documentation

15.200.2.1 const u8* eap_get_identity (struct eap_sm * *sm*, size_t * *len*)

Get the user identity (from EAP-Response/Identity).

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

len Buffer for returning identity length

Returns:

Pointer to the user identity or NULL if not available

15.200.2.2 struct eap_eapol_interface* eap_get_interface (struct eap_sm * *sm*) [read]

Get pointer to EAP-EAPOL interface data.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

Returns:

Pointer to the EAP-EAPOL interface data

15.200.2.3 void eap_server_sm_deinit (struct eap_sm * *sm*)

Deinitialize and free an EAP server state machine.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

This function deinitializes EAP state machine and frees all allocated resources.

15.200.2.4 struct eap_sm* eap_server_sm_init (void * *eapol_ctx*, struct eapol_callbacks * *eapol_cb*, struct eap_config * *conf*) [read]

Allocate and initialize EAP server state machine.

Parameters:

eapol_ctx Context data to be used with eapol_cb calls

eapol_cb Pointer to EAPOL callback functions

conf EAP configuration

Returns:

Pointer to the allocated EAP state machine or NULL on failure

This function allocates and initializes an EAP state machine.

15.200.2.5 int eap_server_sm_step (struct eap_sm * *sm*)

Step EAP server state machine.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_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.

15.200.2.6 int eap_sm_method_pending (struct eap_sm * *sm*)

Query whether EAP method is waiting for pending data.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

Returns:

1 if method is waiting for pending data or 0 if not

15.200.2.7 void eap_sm_notify_cached (struct eap_sm * *sm*)

Notify EAP state machine of cached PMK.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

This function is called when PMKSA caching is used to skip EAP authentication.

15.200.2.8 void eap_sm_pending_cb (struct eap_sm * *sm*)

EAP state machine callback for a pending EAP request.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

This function is called when data for a pending EAP-Request is received.

15.201 src/eap_peer/eap_aka.c File Reference

EAP peer method: EAP-AKA (RFC 4187) and EAP-AKA' (draft-arkko-eap-aka-kdf). `#include "includes.h"`

```
#include "common.h"
#include "eap_peer/eap_i.h"
#include "pcsc_funcs.h"
#include "eap_common/eap_sim_common.h"
#include "sha1.h"
#include "sha256.h"
#include "crypto.h"
#include "eap_peer/eap_config.h"
```

Data Structures

- struct eap_aka_data

Defines

- #define **CLEAR_PSEUDONYM** 0x01
- #define **CLEAR_REAUTH_ID** 0x02
- #define **CLEAR_EAP_ID** 0x04

Functions

- int **eap_peer_aka_register** (void)

15.201.1 Detailed Description

EAP peer method: EAP-AKA (RFC 4187) and EAP-AKA' (draft-arkko-eap-aka-kdf).

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.202 src/eap_server/eap_aka.c File Reference

hostapd / EAP-AKA (RFC 4187) and EAP-AKA' (draft-arkko-eap-aka-kdf) `#include "includes.h"`

```
#include "common.h"
#include "eap_server/eap_i.h"
#include "eap_common/eap_sim_common.h"
#include "eap_server/eap_sim_db.h"
#include "sha1.h"
#include "sha256.h"
#include "crypto.h"
```

Data Structures

- struct eap_aka_data

Functions

- int **eap_server_aka_register** (void)

15.202.1 Detailed Description

hostapd / EAP-AKA (RFC 4187) and EAP-AKA' (draft-arkko-eap-aka-kdf)

Copyright

15.203 src/eap_peer/eap_config.h File Reference

EAP peer configuration data.

Data Structures

- struct eap_peer_config

  EAP peer configuration/credentials.

- struct wpa_config_blob

  Named configuration blob.

Defines

- #define **EAP_CONFIG_FLAGS_PASSWORD_NTHASH** BIT(0)

15.203.1 Detailed Description

EAP peer configuration data.

Copyright

15.204 src/eap_peer/eap_fast.c File Reference

EAP peer method: EAP-FAST (RFC 4851). `#include "includes.h"`

```
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "eap_config.h"
#include "tls.h"
#include "eap_common/eap_tlv_common.h"
#include "sha1.h"
#include "eap_fast_pac.h"
```

Data Structures

- struct eap_fast_data

Defines

- #define **EAP_FAST_PROV_UNAUTH** 1
- #define **EAP_FAST_PROV_AUTH** 2

Functions

- int **eap_peer_fast_register** (void)

15.204.1 Detailed Description

EAP peer method: EAP-FAST (RFC 4851).

Copyright

15.205 src/eap_server/eap_fast.c File Reference

EAP-FAST server (RFC 4851). `#include "includes.h"`

```
#include "common.h"
#include "aes_wrap.h"
#include "sha1.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "tls.h"
#include "eap_common/eap_tlv_common.h"
#include "eap_common/eap_fast_common.h"
```

Data Structures

- struct eap_fast_data

Defines

- #define **PAC_OPAQUE_TYPE_PAD** 0
- #define **PAC_OPAQUE_TYPE_KEY** 1
- #define **PAC_OPAQUE_TYPE_LIFETIME** 2
- #define **PAC_OPAQUE_TYPE_IDENTITY** 3

Functions

- int **eap_server_fast_register** (void)

15.205.1 Detailed Description

EAP-FAST server (RFC 4851).

Copyright

15.206 src/eap_peer/eap_fast_pac.c File Reference

EAP peer method: EAP-FAST PAC file processing. `#include "includes.h"`

```
#include "common.h"
#include "eap_config.h"
#include "eap_i.h"
#include "eap_fast_pac.h"
```

Data Structures

- struct eap_fast_read_ctx

Defines

- #define **EAP_FAST_PAC_BINARY_MAGIC** 0x6ae4920c
- #define **EAP_FAST_PAC_BINARY_FORMAT_VERSION** 0

Functions

- void eap_fast_free_pac (struct eap_fast_pac *pac)

  Free PAC data.

- struct eap_fast_pac * eap_fast_get_pac (struct eap_fast_pac *pac_root, const u8 *a_id, size_t a_id_len, u16 pac_type)

  Get a PAC entry based on A-ID.

- int eap_fast_add_pac (struct eap_fast_pac **pac_root, struct eap_fast_pac **pac_current, struct eap_fast_pac *entry)

  Add a copy of a PAC entry to a list.

- int eap_fast_load_pac (struct eap_sm *sm, struct eap_fast_pac **pac_root, const char *pac_file)

  Load PAC entries (text format).

- int eap_fast_save_pac (struct eap_sm *sm, struct eap_fast_pac *pac_root, const char *pac_file)

  Save PAC entries (text format).

- size_t eap_fast_pac_list_truncate (struct eap_fast_pac *pac_root, size_t max_len)

  Truncate a PAC list to the given length.

- int eap_fast_load_pac_bin (struct eap_sm *sm, struct eap_fast_pac **pac_root, const char *pac_file)

  Load PAC entries (binary format).

- int eap_fast_save_pac_bin (struct eap_sm *sm, struct eap_fast_pac *pac_root, const char *pac_file)

  Save PAC entries (binary format).

15.206.1 Detailed Description

EAP peer method: EAP-FAST PAC file processing.

Copyright

Copyright (c) 2004-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.206.2 Function Documentation

15.206.2.1 int eap_fast_add_pac (struct eap_fast_pac ** *pac_root*, struct eap_fast_pac ** *pac_current*, struct eap_fast_pac * *entry*)

Add a copy of a PAC entry to a list.

Parameters:

pac_root Pointer to PAC list root pointer

pac_current Pointer to the current PAC pointer

entry New entry to clone and add to the list

Returns:

0 on success, -1 on failure

This function makes a clone of the given PAC entry and adds this copied entry to the list (pac_root). If an old entry for the same A-ID is found, it will be removed from the PAC list and in this case, pac_current entry is set to NULL if it was the removed entry.

15.206.2.2 void eap_fast_free_pac (struct eap_fast_pac * *pac*)

Free PAC data.

Parameters:

pac Pointer to the PAC entry

Note that the PAC entry must not be in a list since this function does not remove the list links.

15.206.2.3 struct eap_fast_pac* eap_fast_get_pac (struct eap_fast_pac * *pac_root*, const u8 * *a_id*, size_t *a_id_len*, u16 *pac_type*) [read]

Get a PAC entry based on A-ID.

Parameters:

pac_root Pointer to root of the PAC list

a_id A-ID to search for

a_id_len Length of A-ID

pac_type PAC-Type to search for

Returns:

Pointer to the PAC entry, or NULL if A-ID not found

15.206.2.4 int eap_fast_load_pac (struct eap_sm * *sm*, struct eap_fast_pac ** *pac_root*, const char * *pac_file*)

Load PAC entries (text format).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

pac_root Pointer to root of the PAC list (to be filled)

pac_file Name of the PAC file/blob to load

Returns:

0 on success, -1 on failure

15.206.2.5 int eap_fast_load_pac_bin (struct eap_sm * *sm*, struct eap_fast_pac ** *pac_root*, const char * *pac_file*)

Load PAC entries (binary format).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

pac_root Pointer to root of the PAC list (to be filled)

pac_file Name of the PAC file/blob to load

Returns:

0 on success, -1 on failure

15.206.2.6 size_t eap_fast_pac_list_truncate (struct eap_fast_pac * *pac_root*, size_t *max_len*)

Truncate a PAC list to the given length.

Parameters:

pac_root Root of the PAC list

max_len Maximum length of the list (>= 1)

Returns:

Number of PAC entries removed

15.206.2.7 int eap_fast_save_pac (struct eap_sm * *sm*, struct eap_fast_pac * *pac_root*, const char * *pac_file*)

Save PAC entries (text format).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

pac_root Root of the PAC list

pac_file Name of the PAC file/blob

Returns:

0 on success, -1 on failure

15.206.2.8 int eap_fast_save_pac_bin (struct eap_sm * *sm*, struct eap_fast_pac * *pac_root*, const char * *pac_file*)

Save PAC entries (binary format).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

pac_root Root of the PAC list

pac_file Name of the PAC file/blob

Returns:

0 on success, -1 on failure

15.207 src/eap_peer/eap_fast_pac.h File Reference

EAP peer method: EAP-FAST PAC file processing. `#include "eap_common/eap_fast_common.h"`

Data Structures

- struct eap_fast_pac

Functions

- void eap_fast_free_pac (struct eap_fast_pac *pac)

  Free PAC data.

- struct eap_fast_pac * eap_fast_get_pac (struct eap_fast_pac *pac_root, const u8 *a_id, size_t a_id_len, u16 pac_type)

  Get a PAC entry based on A-ID.

- int eap_fast_add_pac (struct eap_fast_pac **pac_root, struct eap_fast_pac **pac_current, struct eap_fast_pac *entry)

  Add a copy of a PAC entry to a list.

- int eap_fast_load_pac (struct eap_sm *sm, struct eap_fast_pac **pac_root, const char *pac_file)

  Load PAC entries (text format).

- int eap_fast_save_pac (struct eap_sm *sm, struct eap_fast_pac *pac_root, const char *pac_file)

  Save PAC entries (text format).

- size_t eap_fast_pac_list_truncate (struct eap_fast_pac *pac_root, size_t max_len)

  Truncate a PAC list to the given length.

- int eap_fast_load_pac_bin (struct eap_sm *sm, struct eap_fast_pac **pac_root, const char *pac_file)

  Load PAC entries (binary format).

- int eap_fast_save_pac_bin (struct eap_sm *sm, struct eap_fast_pac *pac_root, const char *pac_file)

  Save PAC entries (binary format).

15.207.1 Detailed Description

EAP peer method: EAP-FAST PAC file processing.

Copyright

15.207.2 Function Documentation

15.207.2.1 int eap_fast_add_pac (struct eap_fast_pac ** *pac_root*, struct eap_fast_pac ** *pac_current*, struct eap_fast_pac * *entry*)

Add a copy of a PAC entry to a list.

Parameters:

pac_root Pointer to PAC list root pointer

pac_current Pointer to the current PAC pointer

entry New entry to clone and add to the list

Returns:

0 on success, -1 on failure

This function makes a clone of the given PAC entry and adds this copied entry to the list (pac_root). If an old entry for the same A-ID is found, it will be removed from the PAC list and in this case, pac_current entry is set to NULL if it was the removed entry.

15.207.2.2 void eap_fast_free_pac (struct eap_fast_pac * *pac*)

Free PAC data.

Parameters:

pac Pointer to the PAC entry

Note that the PAC entry must not be in a list since this function does not remove the list links.

15.207.2.3 struct eap_fast_pac* eap_fast_get_pac (struct eap_fast_pac * *pac_root*, const u8 * *a_id*, size_t *a_id_len*, u16 *pac_type*) [read]

Get a PAC entry based on A-ID.

Parameters:

pac_root Pointer to root of the PAC list

a_id A-ID to search for

a_id_len Length of A-ID

pac_type PAC-Type to search for

Returns:

Pointer to the PAC entry, or NULL if A-ID not found

15.207.2.4 int eap_fast_load_pac (struct eap_sm * *sm*, struct eap_fast_pac ** *pac_root*, const char * *pac_file*)

Load PAC entries (text format).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

pac_root Pointer to root of the PAC list (to be filled)

pac_file Name of the PAC file/blob to load

Returns:

0 on success, -1 on failure

15.207.2.5 int eap_fast_load_pac_bin (struct eap_sm * *sm*, struct eap_fast_pac ** *pac_root*, const char * *pac_file*)

Load PAC entries (binary format).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

pac_root Pointer to root of the PAC list (to be filled)

pac_file Name of the PAC file/blob to load

Returns:

0 on success, -1 on failure

15.207.2.6 size_t eap_fast_pac_list_truncate (struct eap_fast_pac * *pac_root*, size_t *max_len*)

Truncate a PAC list to the given length.

Parameters:

pac_root Root of the PAC list

max_len Maximum length of the list (>= 1)

Returns:

Number of PAC entries removed

15.207.2.7 int eap_fast_save_pac (struct eap_sm * *sm*, struct eap_fast_pac * *pac_root*, const char * *pac_file*)

Save PAC entries (text format).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

pac_root Root of the PAC list

pac_file Name of the PAC file/blob

Returns:

0 on success, -1 on failure

15.207.2.8 int eap_fast_save_pac_bin (struct eap_sm * *sm*, struct eap_fast_pac * *pac_root*, const char * *pac_file*)

Save PAC entries (binary format).

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

pac_root Root of the PAC list

pac_file Name of the PAC file/blob

Returns:

0 on success, -1 on failure

15.208 src/eap_peer/eap_gpsk.c File Reference

EAP peer method: EAP-GPSK (RFC 5433). `#include "includes.h"`

`#include "common.h"`

`#include "eap_peer/eap_i.h"`

`#include "eap_common/eap_gpsk_common.h"`

Data Structures

- struct eap_gpsk_data

Functions

- int **eap_peer_gpsk_register** (void)

15.208.1 Detailed Description

EAP peer method: EAP-GPSK (RFC 5433).

Copyright

15.209 src/eap_server/eap_gpsk.c File Reference

hostapd / EAP-GPSK (RFC 5433) server `#include "includes.h"`

`#include "common.h"`

`#include "eap_server/eap_i.h"`

`#include "eap_common/eap_gpsk_common.h"`

Data Structures

- struct eap_gpsk_data

Defines

- #define **MAX_NUM_CSUITES** 2

Functions

- int **eap_server_gpsk_register** (void)

15.209.1 Detailed Description

hostapd / EAP-GPSK (RFC 5433) server

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.210 src/eap_peer/eap_gtc.c File Reference

EAP peer method: EAP-GTC (RFC 3748). `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

Data Structures

- struct eap_gtc_data

Functions

- int **eap_peer_gtc_register** (void)

15.210.1 Detailed Description

EAP peer method: EAP-GTC (RFC 3748).

Copyright

15.211 src/eap_server/eap_gtc.c File Reference

hostapd / EAP-GTC (RFC 3748) `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

Data Structures

- struct eap_gtc_data

Functions

- int **eap_server_gtc_register** (void)

15.211.1 Detailed Description

hostapd / EAP-GTC (RFC 3748)

Copyright

15.212 src/eap_peer/eap_i.h File Reference

EAP peer state machines internal structures (RFC 4137). `#include "wpabuf.h"`

`#include "eap_peer/eap.h"`

`#include "eap_common/eap_common.h"`

Data Structures

- struct eap_method_ret

  EAP return values from struct eap_method::process().

- struct eap_method

  EAP method interface.

- struct eap_sm

  EAP state machine data.

Defines

- #define **EAP_PEER_METHOD_INTERFACE_VERSION** 1

Enumerations

- enum **EapDecision** { **DECISION_FAIL**, **DECISION_COND_SUCC**, **DECISION_UNCOND_-SUCC** }
- enum **EapMethodState** {

  METHOD_NONE, **METHOD_INIT**, **METHOD_CONT**, **METHOD_MAY_CONT**,

  METHOD_DONE }

Functions

- const u8 * eap_get_config_identity (struct eap_sm *sm, size_t *len)

  Get identity from the network configuration.

- const u8 * eap_get_config_password (struct eap_sm *sm, size_t *len)

  Get password from the network configuration.

- const u8 * eap_get_config_password2 (struct eap_sm *sm, size_t *len, int *hash)

  Get password from the network configuration.

- const u8 * eap_get_config_new_password (struct eap_sm *sm, size_t *len)

  Get new password from network configuration.

- const u8 * eap_get_config_otp (struct eap_sm *sm, size_t *len)

  Get one-time password from the network configuration.

- void eap_clear_config_otp (struct eap_sm *sm)
  Clear used one-time password.
- const char * eap_get_config_phase1 (struct eap_sm *sm)
  Get phase1 data from the network configuration.
- const char * eap_get_config_phase2 (struct eap_sm *sm)
  Get phase2 data from the network configuration.
- struct eap_peer_config * eap_get_config (struct eap_sm *sm)
  Get current network configuration.
- void eap_set_config_blob (struct eap_sm *sm, struct wpa_config_blob *blob)
  Set or add a named configuration blob.
- struct wpa_config_blob * eap_get_config_blob (struct eap_sm *sm, const char *name)
  Get a named configuration blob.
- void eap_notify_pending (struct eap_sm *sm)
  Notify that EAP method is ready to re-process a request.
- int eap_allowed_method (struct eap_sm *sm, int vendor, u32 method)
  Check whether EAP method is allowed.

15.212.1 Detailed Description

EAP peer state machines internal structures (RFC 4137).

Copyright

15.212.2 Function Documentation

15.212.2.1 int eap_allowed_method (struct eap_sm * *sm*, int *vendor*, u32 *method*)

Check whether EAP method is allowed.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

vendor Vendor-Id for expanded types or 0 = IETF for legacy types

method EAP type

Returns:

1 = allowed EAP method, 0 = not allowed

15.212.2.2 void eap_clear_config_otp (struct eap_sm * *sm*)

Clear used one-time password.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

This function clears a used one-time password (OTP) from the current network configuration. This should be called when the OTP has been used and is not needed anymore.

15.212.2.3 struct eap_peer_config* eap_get_config (struct eap_sm * *sm*) [read]

Get current network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Returns:

Pointer to the current network configuration or NULL if not found

EAP peer methods should avoid using this function if they can use other access functions, like eap_get_config_identity() and eap_get_config_password(), that do not require direct access to struct eap_peer_config.

15.212.2.4 struct wpa_config_blob* eap_get_config_blob (struct eap_sm * *sm*, const char * *name*) [read]

Get a named configuration blob.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

name Name of the blob

Returns:

Pointer to blob data or NULL if not found

15.212.2.5 const u8* eap_get_config_identity (struct eap_sm * *sm*, size_t * *len*)

Get identity from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the identity

Returns:

Pointer to the identity or NULL if not found

15.212.2.6 const u8* eap_get_config_new_password (struct eap_sm * *sm*, size_t * *len*)

Get new password from network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the new password

Returns:

Pointer to the new password or NULL if not found

15.212.2.7 const u8* eap_get_config_otp (struct eap_sm * *sm*, size_t * *len*)

Get one-time password from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the one-time password

Returns:

Pointer to the one-time password or NULL if not found

15.212.2.8 const u8* eap_get_config_password (struct eap_sm * *sm*, size_t * *len*)

Get password from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the password

Returns:

Pointer to the password or NULL if not found

15.212.2.9 const u8* eap_get_config_password2 (struct eap_sm * *sm*, size_t * *len*, int * *hash*)

Get password from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

len Buffer for the length of the password

hash Buffer for returning whether the password is stored as a NtPasswordHash instead of plaintext password; can be NULL if this information is not needed

Returns:

Pointer to the password or NULL if not found

15.212.2.10 const char* eap_get_config_phase1 (struct eap_sm * *sm*)

Get phase1 data from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Returns:

Pointer to the phase1 data or NULL if not found

15.212.2.11 const char* eap_get_config_phase2 (struct eap_sm * *sm*)

Get phase2 data from the network configuration.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

Returns:

Pointer to the phase1 data or NULL if not found

15.212.2.12 void eap_notify_pending (struct eap_sm * *sm*)

Notify that EAP method is ready to re-process a request.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

An EAP method can perform a pending operation (e.g., to get a response from an external process). Once the response is available, this function can be used to request EAPOL state machine to retry delivering the previously received (and still unanswered) EAP request to EAP state machine.

15.212.2.13 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_peer_sm_init()

blob New value for the blob

Adds a new configuration blob or replaces the current value of an existing blob.

15.213 src/eap_server/eap_i.h File Reference

hostapd / EAP Authenticator state machine internal structures (RFC 4137) `#include "wpabuf.h"`

`#include "eap_server/eap.h"`

`#include "eap_common/eap_common.h"`

Data Structures

- struct eap_method

  EAP method interface.

- struct eap_sm

  EAP state machine data.

Defines

- #define **EAP_SERVER_METHOD_INTERFACE_VERSION** 1

Functions

- int eap_user_get (struct eap_sm *sm, const u8 *identity, size_t identity_len, int phase2)

  Fetch user information from the database.

- void eap_sm_process_nak (struct eap_sm *sm, const u8 *nak_list, size_t len)

  Process EAP-Response/Nak.

15.213.1 Detailed Description

hostapd / EAP Authenticator state machine internal structures (RFC 4137)

Copyright

15.213.2 Function Documentation

15.213.2.1 void eap_sm_process_nak (struct eap_sm * *sm*, const u8 * *nak_list*, size_t *len*)

Process EAP-Response/Nak.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

nak_list Nak list (allowed methods) from the supplicant

len Length of nak_list in bytes

This function is called when EAP-Response/Nak is received from the supplicant. This can happen for both phase 1 and phase 2 authentications.

15.213.2.2 int eap_user_get (struct eap_sm * *sm*, const u8 * *identity*, size_t *identity_len*, int *phase2*)

Fetch user information from the database.

Parameters:

sm Pointer to EAP state machine allocated with eap_server_sm_init()

identity Identity (User-Name) of the user

identity_len Length of identity in bytes

phase2 0 = EAP phase1 user, 1 = EAP phase2 (tunneled) user

Returns:

0 on success, or -1 on failure

This function is used to fetch user information for EAP. The user will be selected based on the specified identity. sm->user and sm->user_eap_method_index are updated for the new user when a matching user is found. sm->user can be used to get user information (e.g., password).

15.214 src/eap_peer/eap_ikev2.c File Reference

EAP-IKEv2 peer (RFC 5106). `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

`#include "eap_common/eap_ikev2_common.h"`

`#include "ikev2.h"`

Data Structures

- struct eap_ikev2_data

Functions

- int **eap_peer_ikev2_register** (void)

15.214.1 Detailed Description

EAP-IKEv2 peer (RFC 5106).

Copyright

15.215 src/eap_server/eap_ikev2.c File Reference

EAP-IKEv2 server (RFC 5106). `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

`#include "eap_common/eap_ikev2_common.h"`

`#include "ikev2.h"`

Data Structures

- struct eap_ikev2_data

Functions

- int **eap_server_ikev2_register** (void)

15.215.1 Detailed Description

EAP-IKEv2 server (RFC 5106).

Copyright

15.216 src/eap_peer/eap_leap.c File Reference

EAP peer method: LEAP. `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

`#include "ms_funcs.h"`

`#include "crypto.h"`

Data Structures

- struct eap_leap_data

Defines

- #define **LEAP_VERSION** 1
- #define **LEAP_CHALLENGE_LEN** 8
- #define **LEAP_RESPONSE_LEN** 24
- #define **LEAP_KEY_LEN** 16

Functions

- int **eap_peer_leap_register** (void)

15.216.1 Detailed Description

EAP peer method: LEAP.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.217 src/eap_peer/eap_md5.c File Reference

EAP peer method: EAP-MD5 (RFC 3748 and RFC 1994). `#include "includes.h"`

```
#include "common.h"
#include "eap_i.h"
#include "eap_common/chap.h"
```

Functions

- int **eap_peer_md5_register** (void)

15.217.1 Detailed Description

EAP peer method: EAP-MD5 (RFC 3748 and RFC 1994).

Copyright

15.218 src/eap_server/eap_md5.c File Reference

hostapd / EAP-MD5 server `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

`#include "eap_common/chap.h"`

Data Structures

- struct eap_md5_data

Defines

- #define **CHALLENGE_LEN** 16

Functions

- int **eap_server_md5_register** (void)

15.218.1 Detailed Description

hostapd / EAP-MD5 server

Copyright

15.219 src/eap_peer/eap_methods.c File Reference

EAP peer: Method registration. `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

`#include "eap_methods.h"`

Functions

- struct eap_method * eap_peer_get_eap_method (int vendor, EapType method)

  Get EAP method based on type number.

- EapType eap_peer_get_type (const char *name, int *vendor)

  Get EAP type for the given EAP method name.

- const char * eap_get_name (int vendor, EapType type)

  Get EAP method name for the given EAP type.

- size_t eap_get_names (char *buf, size_t buflen)

  Get space separated list of names for supported EAP methods.

- char ** eap_get_names_as_string_array (size_t *num)

  Get supported EAP methods as string array.

- struct eap_method * eap_peer_get_methods (size_t *count)

  Get a list of enabled EAP peer methods.

- struct eap_method * eap_peer_method_alloc (int version, int vendor, EapType method, const char *name)

  Allocate EAP peer method structure.

- void eap_peer_method_free (struct eap_method *method)

  Free EAP peer method structure.

- int eap_peer_method_register (struct eap_method *method)

  Register an EAP peer method.

- int eap_peer_register_methods (void)

  Register statically linked EAP peer methods.

- void eap_peer_unregister_methods (void)

  Unregister EAP peer methods.

15.219.1 Detailed Description

EAP peer: Method registration.

Copyright

15.219.2 Function Documentation

15.219.2.1 const char* eap_get_name (int *vendor*, EapType *type*)

Get EAP method name for the given EAP type.

Parameters:

vendor EAP Vendor-Id (0 = IETF)

type EAP method type

Returns:

EAP method name, e.g., TLS, or NULL if not found

This function maps EAP type numbers into EAP type names based on the list of EAP methods included in the build.

15.219.2.2 size_t eap_get_names (char * *buf*, size_t *buflen*)

Get space separated list of names for supported EAP methods.

Parameters:

buf Buffer for names

buflen Buffer length

Returns:

Number of characters written into buf (not including nul termination)

15.219.2.3 char** eap_get_names_as_string_array (size_t * *num*)

Get supported EAP methods as string array.

Parameters:

num Buffer for returning the number of items in array, not including NULL terminator. This parameter can be NULL if the length is not needed.

Returns:

A NULL-terminated array of strings, or NULL on error.

This function returns the list of names for all supported EAP methods as an array of strings. The caller must free the returned array items and the array.

15.219.2.4 struct eap_method* eap_peer_get_eap_method (int *vendor*, EapType *method*) [read]

Get EAP method based on type number.

Parameters:

vendor EAP Vendor-Id (0 = IETF)

method EAP type number

Returns:

Pointer to EAP method or NULL if not found

15.219.2.5 struct eap_method* eap_peer_get_methods (size_t * *count*) [read]

Get a list of enabled EAP peer methods.

Parameters:

count Set to number of available methods

Returns:

List of enabled EAP peer methods

15.219.2.6 EapType eap_peer_get_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP method name.

Parameters:

name EAP method name, e.g., TLS

vendor Buffer for returning EAP Vendor-Id

Returns:

EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers based on the list of EAP methods included in the build.

15.219.2.7 struct eap_method* eap_peer_method_alloc (int *version*, int *vendor*, EapType *method*, const char * *name*) [read]

Allocate EAP peer method structure.

Parameters:

version Version of the EAP peer method interface (set to EAP_PEER_METHOD_INTERFACE_VERSION)

vendor EAP Vendor-ID (EAP_VENDOR_*) (0 = IETF)

method EAP type number (EAP_TYPE_∗)

name Name of the method (e.g., "TLS")

Returns:

Allocated EAP method structure or NULL on failure

The returned structure should be freed with eap_peer_method_free() when it is not needed anymore.

15.219.2.8 void eap_peer_method_free (struct eap_method ∗ *method*)

Free EAP peer method structure.

Parameters:

method Method structure allocated with eap_peer_method_alloc()

15.219.2.9 int eap_peer_method_register (struct eap_method ∗ *method*)

Register an EAP peer method.

Parameters:

method EAP method to register

Returns:

0 on success, -1 on invalid method, or -2 if a matching EAP method has already been registered

Each EAP peer method needs to call this function to register itself as a supported EAP method.

15.219.2.10 int eap_peer_register_methods (void)

Register statically linked EAP peer methods.

Returns:

0 on success, -1 on failure

This function is called at program initialization to register all EAP peer methods that were linked in statically.

15.219.2.11 void eap_peer_unregister_methods (void)

Unregister EAP peer methods. This function is called at program termination to unregister all EAP peer methods.

15.220 src/eap_server/eap_methods.c File Reference

hostapd / EAP method registration `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

`#include "eap_methods.h"`

Functions

- struct eap_method * eap_server_get_eap_method (int vendor, EapType method)

  Get EAP method based on type number.

- EapType eap_server_get_type (const char *name, int *vendor)

  Get EAP type for the given EAP method name.

- struct eap_method * eap_server_method_alloc (int version, int vendor, EapType method, const char *name)

  Allocate EAP server method structure.

- void eap_server_method_free (struct eap_method *method)

  Free EAP server method structure.

- int eap_server_method_register (struct eap_method *method)

  Register an EAP server method.

- int eap_server_register_methods (void)

  Register statically linked EAP server methods.

- void eap_server_unregister_methods (void)

  Unregister EAP server methods.

15.220.1 Detailed Description

hostapd / EAP method registration

Copyright

Copyright (c) 2004-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.220.2 Function Documentation

15.220.2.1 struct eap_method* eap_server_get_eap_method (int *vendor*, EapType *method*) [read]

Get EAP method based on type number.

Parameters:

vendor EAP Vendor-Id (0 = IETF)

method EAP type number

Returns:

Pointer to EAP method or NULL if not found

15.220.2.2 EapType eap_server_get_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP method name.

Parameters:

name EAP method name, e.g., TLS

vendor Buffer for returning EAP Vendor-Id

Returns:

EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers based on the list of EAP methods included in the build.

15.220.2.3 struct eap_method* eap_server_method_alloc (int *version*, int *vendor*, EapType *method*, const char * *name*) [read]

Allocate EAP server method structure.

Parameters:

version Version of the EAP server method interface (set to EAP_SERVER_METHOD_INTERFACE_VERSION)

vendor EAP Vendor-ID (EAP_VENDOR_*) (0 = IETF)

method EAP type number (EAP_TYPE_*)

name Name of the method (e.g., "TLS")

Returns:

Allocated EAP method structure or NULL on failure

The returned structure should be freed with eap_server_method_free() when it is not needed anymore.

15.220.2.4 void eap_server_method_free (struct eap_method * *method*)

Free EAP server method structure.

Parameters:

method Method structure allocated with eap_server_method_alloc()

15.220.2.5 int eap_server_method_register (struct eap_method * *method*)

Register an EAP server method.

Parameters:

method EAP method to register

Returns:

0 on success, -1 on invalid method, or -2 if a matching EAP method has already been registered

Each EAP server method needs to call this function to register itself as a supported EAP method.

15.220.2.6 int eap_server_register_methods (void)

Register statically linked EAP server methods.

Returns:

0 on success, -1 on failure

This function is called at program initialization to register all EAP server methods that were linked in statically.

15.220.2.7 void eap_server_unregister_methods (void)

Unregister EAP server methods. This function is called at program termination to unregister all EAP server methods.

15.221 src/eap_peer/eap_methods.h File Reference

EAP peer: Method registration. `#include "eap_common/eap_defs.h"`

Functions

- struct eap_method * eap_peer_get_eap_method (int vendor, EapType method)

  Get EAP method based on type number.

- struct eap_method * eap_peer_get_methods (size_t *count)

  Get a list of enabled EAP peer methods.

- struct eap_method * eap_peer_method_alloc (int version, int vendor, EapType method, const char *name)

  Allocate EAP peer method structure.

- void eap_peer_method_free (struct eap_method *method)

  Free EAP peer method structure.

- int eap_peer_method_register (struct eap_method *method)

  Register an EAP peer method.

- EapType eap_peer_get_type (const char *name, int *vendor)

  Get EAP type for the given EAP method name.

- const char * eap_get_name (int vendor, EapType type)

  Get EAP method name for the given EAP type.

- size_t eap_get_names (char *buf, size_t buflen)

  Get space separated list of names for supported EAP methods.

- char ** eap_get_names_as_string_array (size_t *num)

  Get supported EAP methods as string array.

- int eap_peer_register_methods (void)

  Register statically linked EAP peer methods.

- void eap_peer_unregister_methods (void)

  Unregister EAP peer methods.

15.221.1 Detailed Description

EAP peer: Method registration.

Copyright

15.221.2 Function Documentation

15.221.2.1 const char* eap_get_name (int *vendor*, EapType *type*)

Get EAP method name for the given EAP type.

Parameters:

vendor EAP Vendor-Id (0 = IETF)

type EAP method type

Returns:

EAP method name, e.g., TLS, or NULL if not found

This function maps EAP type numbers into EAP type names based on the list of EAP methods included in the build.

15.221.2.2 size_t eap_get_names (char * *buf*, size_t *buflen*)

Get space separated list of names for supported EAP methods.

Parameters:

buf Buffer for names

buflen Buffer length

Returns:

Number of characters written into buf (not including nul termination)

15.221.2.3 char** eap_get_names_as_string_array (size_t * *num*)

Get supported EAP methods as string array.

Parameters:

num Buffer for returning the number of items in array, not including NULL terminator. This parameter can be NULL if the length is not needed.

Returns:

A NULL-terminated array of strings, or NULL on error.

This function returns the list of names for all supported EAP methods as an array of strings. The caller must free the returned array items and the array.

15.221.2.4 struct eap_method* eap_peer_get_eap_method (int *vendor*, EapType *method*) [read]

Get EAP method based on type number.

Parameters:

vendor EAP Vendor-Id (0 = IETF)

method EAP type number

Returns:

Pointer to EAP method or NULL if not found

15.221.2.5 struct eap_method* eap_peer_get_methods (size_t * *count*) [read]

Get a list of enabled EAP peer methods.

Parameters:

count Set to number of available methods

Returns:

List of enabled EAP peer methods

15.221.2.6 EapType eap_peer_get_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP method name.

Parameters:

name EAP method name, e.g., TLS

vendor Buffer for returning EAP Vendor-Id

Returns:

EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers based on the list of EAP methods included in the build.

15.221.2.7 struct eap_method* eap_peer_method_alloc (int *version*, int *vendor*, EapType *method*, const char * *name*) [read]

Allocate EAP peer method structure.

Parameters:

version Version of the EAP peer method interface (set to EAP_PEER_METHOD_INTERFACE_VERSION)

vendor EAP Vendor-ID (EAP_VENDOR_*) (0 = IETF)

method EAP type number (EAP_TYPE_∗)

name Name of the method (e.g., "TLS")

Returns:

Allocated EAP method structure or NULL on failure

The returned structure should be freed with eap_peer_method_free() when it is not needed anymore.

15.221.2.8 void eap_peer_method_free (struct eap_method ∗ *method*)

Free EAP peer method structure.

Parameters:

method Method structure allocated with eap_peer_method_alloc()

15.221.2.9 int eap_peer_method_register (struct eap_method ∗ *method*)

Register an EAP peer method.

Parameters:

method EAP method to register

Returns:

0 on success, -1 on invalid method, or -2 if a matching EAP method has already been registered

Each EAP peer method needs to call this function to register itself as a supported EAP method.

15.221.2.10 int eap_peer_register_methods (void)

Register statically linked EAP peer methods.

Returns:

0 on success, -1 on failure

This function is called at program initialization to register all EAP peer methods that were linked in statically.

15.221.2.11 void eap_peer_unregister_methods (void)

Unregister EAP peer methods. This function is called at program termination to unregister all EAP peer methods.

15.222 src/eap_server/eap_methods.h File Reference

hostapd / EAP method registration

Functions

- struct eap_method * eap_server_get_eap_method (int vendor, EapType method)

  Get EAP method based on type number.

- struct eap_method * eap_server_method_alloc (int version, int vendor, EapType method, const char *name)

  Allocate EAP server method structure.

- void eap_server_method_free (struct eap_method *method)

  Free EAP server method structure.

- int eap_server_method_register (struct eap_method *method)

  Register an EAP server method.

- EapType eap_server_get_type (const char *name, int *vendor)

  Get EAP type for the given EAP method name.

- int eap_server_register_methods (void)

  Register statically linked EAP server methods.

- void eap_server_unregister_methods (void)

  Unregister EAP server methods.

15.222.1 Detailed Description

hostapd / EAP method registration

Copyright

Copyright (c) 2004-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.222.2 Function Documentation

15.222.2.1 struct eap_method* eap_server_get_eap_method (int *vendor*, EapType *method*) [read]

Get EAP method based on type number.

Parameters:

vendor EAP Vendor-Id (0 = IETF)

method EAP type number

Returns:

Pointer to EAP method or NULL if not found

15.222.2.2 EapType eap_server_get_type (const char * *name*, int * *vendor*)

Get EAP type for the given EAP method name.

Parameters:

name EAP method name, e.g., TLS

vendor Buffer for returning EAP Vendor-Id

Returns:

EAP method type or EAP_TYPE_NONE if not found

This function maps EAP type names into EAP type numbers based on the list of EAP methods included in the build.

15.222.2.3 struct eap_method* eap_server_method_alloc (int *version*, int *vendor*, EapType *method*, const char * *name*) [read]

Allocate EAP server method structure.

Parameters:

version Version of the EAP server method interface (set to EAP_SERVER_METHOD_INTERFACE_VERSION)

vendor EAP Vendor-ID (EAP_VENDOR_*) (0 = IETF)

method EAP type number (EAP_TYPE_*)

name Name of the method (e.g., "TLS")

Returns:

Allocated EAP method structure or NULL on failure

The returned structure should be freed with eap_server_method_free() when it is not needed anymore.

15.222.2.4 void eap_server_method_free (struct eap_method * *method*)

Free EAP server method structure.

Parameters:

method Method structure allocated with eap_server_method_alloc()

15.222.2.5 int eap_server_method_register (struct eap_method * *method*)

Register an EAP server method.

Parameters:

method EAP method to register

Returns:

0 on success, -1 on invalid method, or -2 if a matching EAP method has already been registered

Each EAP server method needs to call this function to register itself as a supported EAP method.

15.222.2.6 int eap_server_register_methods (void)

Register statically linked EAP server methods.

Returns:

0 on success, -1 on failure

This function is called at program initialization to register all EAP server methods that were linked in statically.

15.222.2.7 void eap_server_unregister_methods (void)

Unregister EAP server methods. This function is called at program termination to unregister all EAP server methods.

15.223 src/eap_peer/eap_mschapv2.c File Reference

EAP peer method: EAP-MSCHAPV2 (draft-kamath-pppext-eap-mschapv2-00.txt). `#include "includes.h"`

```
#include "common.h"
#include "eap_i.h"
#include "eap_config.h"
#include "ms_funcs.h"
#include "wpa_ctrl.h"
#include "mschapv2.h"
```

Data Structures

- struct eap_mschapv2_hdr
- struct ms_response
- struct ms_change_password
- struct eap_mschapv2_data

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 **ERROR_RESTRICTED_LOGON_HOURS** 646
- #define **ERROR_ACCT_DISABLED** 647
- #define **ERROR_PASSWD_EXPIRED** 648
- #define **ERROR_NO_DIALIN_PERMISSION** 649
- #define **ERROR_AUTHENTICATION_FAILURE** 691
- #define **ERROR_CHANGING_PASSWORD** 709
- #define **PASSWD_CHANGE_CHAL_LEN** 16
- #define **MSCHAPV2_KEY_LEN** 16

Functions

- int eap_peer_mschapv2_register (void)

  Register EAP-MSCHAPv2 peer method.

Variables

- struct eap_mschapv2_hdr **STRUCT_PACKED**

15.223.1 Detailed Description

EAP peer method: EAP-MSCHAPV2 (draft-kamath-pppext-eap-mschapv2-00.txt).

Copyright

This file implements EAP peer part of EAP-MSCHAPV2 method (EAP type 26). draft-kamath-pppext-eap-mschapv2-00.txt defines the Microsoft EAP CHAP Extensions Protocol, Version 2, for mutual authentication and key derivation. This encapsulates MS-CHAP-v2 protocol which is defined in RFC 2759. Use of EAP-MSCHAPV2 derived keys with MPPE cipher is described in RFC 3079.

15.223.2 Function Documentation

15.223.2.1 int eap_peer_mschapv2_register (void)

Register EAP-MSCHAPv2 peer method.

Returns:

0 on success, -1 on failure

This function is used to register EAP-MSCHAPv2 peer method into the EAP method list.

15.224 src/eap_server/eap_mschapv2.c File Reference

hostapd / EAP-MSCHAPv2 (draft-kamath-pppext-eap-mschapv2-00.txt) server `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

`#include "ms_funcs.h"`

Data Structures

- struct eap_mschapv2_hdr
- struct eap_mschapv2_data

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
- #define **CHALLENGE_LEN** 16

Functions

- int **eap_server_mschapv2_register** (void)

Variables

- struct eap_mschapv2_hdr **STRUCT_PACKED**

15.224.1 Detailed Description

hostapd / EAP-MSCHAPv2 (draft-kamath-pppext-eap-mschapv2-00.txt) server

Copyright

15.225 src/eap_peer/eap_otp.c File Reference

EAP peer method: EAP-OTP (RFC 3748). `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

Functions

- int **eap_peer_otp_register** (void)

15.225.1 Detailed Description

EAP peer method: EAP-OTP (RFC 3748).

Copyright

15.226 src/eap_peer/eap_pax.c File Reference

EAP peer method: EAP-PAX (RFC 4746). `#include "includes.h"`

```
#include "common.h"
#include "eap_peer/eap_i.h"
#include "eap_common/eap_pax_common.h"
#include "sha1.h"
#include "crypto.h"
```

Data Structures

- struct eap_pax_data

Functions

- int **eap_peer_pax_register** (void)

15.226.1 Detailed Description

EAP peer method: EAP-PAX (RFC 4746).

Copyright

15.227 src/eap_server/eap_pax.c File Reference

hostapd / EAP-PAX (RFC 4746) server `#include "includes.h"`

`#include "common.h"`

`#include "eap_server/eap_i.h"`

`#include "eap_common/eap_pax_common.h"`

Data Structures

- struct eap_pax_data

Functions

- int **eap_server_pax_register** (void)

15.227.1 Detailed Description

hostapd / EAP-PAX (RFC 4746) server

Copyright

15.228 src/eap_peer/eap_peap.c File Reference

EAP peer method: EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-10.txt). `#include "includes.h"`

```
#include "common.h"
#include "crypto/sha1.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "eap_config.h"
#include "tls.h"
#include "eap_common/eap_tlv_common.h"
#include "eap_common/eap_peap_common.h"
#include "tncc.h"
```

Data Structures

- struct eap_peap_data

Defines

- #define **EAP_PEAP_VERSION** 1

Functions

- int **eap_peer_peap_register** (void)

15.228.1 Detailed Description

EAP peer method: EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-10.txt).

Copyright

15.229 src/eap_server/eap_peap.c File Reference

hostapd / EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-10.txt) `#include "includes.h"`

```
#include "common.h"
#include "sha1.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "eap_common/eap_tlv_common.h"
#include "eap_common/eap_peap_common.h"
#include "tls.h"
#include "tncs.h"
```

Data Structures

- struct eap_peap_data

Defines

- #define **EAP_PEAP_VERSION** 1

Functions

- int **eap_server_peap_register** (void)

15.229.1 Detailed Description

hostapd / EAP-PEAP (draft-josefsson-pppext-eap-tls-eap-10.txt)

Copyright

15.230 src/eap_peer/eap_psk.c File Reference

EAP peer method: EAP-PSK (RFC 4764). `#include "includes.h"`

```
#include "common.h"
#include "eap_peer/eap_i.h"
#include "aes_wrap.h"
#include "eap_common/eap_psk_common.h"
```

Data Structures

- struct eap_psk_data

Functions

- int **eap_peer_psk_register** (void)

15.230.1 Detailed Description

EAP peer method: EAP-PSK (RFC 4764).

Copyright

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.

15.231 src/eap_server/eap_psk.c File Reference

hostapd / EAP-PSK (RFC 4764) server `#include "includes.h"`

```
#include "common.h"
#include "eap_server/eap_i.h"
#include "aes_wrap.h"
#include "eap_common/eap_psk_common.h"
```

Data Structures

- struct eap_psk_data

Functions

- int **eap_server_psk_register** (void)

15.231.1 Detailed Description

hostapd / EAP-PSK (RFC 4764) server

Copyright

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.

15.232 src/eap_peer/eap_sake.c File Reference

EAP peer method: EAP-SAKE (RFC 4763). `#include "includes.h"`

`#include "common.h"`

`#include "eap_peer/eap_i.h"`

`#include "eap_common/eap_sake_common.h"`

Data Structures

- struct eap_sake_data

Functions

- int **eap_peer_sake_register** (void)

15.232.1 Detailed Description

EAP peer method: EAP-SAKE (RFC 4763).

Copyright

15.233 src/eap_server/eap_sake.c File Reference

hostapd / EAP-SAKE (RFC 4763) server `#include "includes.h"`

`#include "common.h"`

`#include "eap_server/eap_i.h"`

`#include "eap_common/eap_sake_common.h"`

Data Structures

- struct eap_sake_data

Functions

- int **eap_server_sake_register** (void)

15.233.1 Detailed Description

hostapd / EAP-SAKE (RFC 4763) server

Copyright

15.234 src/eap_peer/eap_sim.c File Reference

EAP peer method: EAP-SIM (RFC 4186). `#include "includes.h"`

```
#include "common.h"
#include "eap_peer/eap_i.h"
#include "eap_config.h"
#include "pcsc_funcs.h"
#include "eap_common/eap_sim_common.h"
```

Data Structures

- struct eap_sim_data

Defines

- #define **CLEAR_PSEUDONYM** 0x01
- #define **CLEAR_REAUTH_ID** 0x02
- #define **CLEAR_EAP_ID** 0x04

Functions

- int **eap_peer_sim_register** (void)

15.234.1 Detailed Description

EAP peer method: EAP-SIM (RFC 4186).

Copyright

15.235 src/eap_server/eap_sim.c File Reference

hostapd / EAP-SIM (RFC 4186) `#include "includes.h"`

`#include "common.h"`

`#include "eap_server/eap_i.h"`

`#include "eap_common/eap_sim_common.h"`

`#include "eap_server/eap_sim_db.h"`

Data Structures

- struct eap_sim_data

Functions

- int **eap_server_sim_register** (void)

15.235.1 Detailed Description

hostapd / EAP-SIM (RFC 4186)

Copyright

15.236 src/eap_peer/eap_tls.c File Reference

EAP peer method: EAP-TLS (RFC 2716). `#include "includes.h"`

```
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "eap_config.h"
#include "tls.h"
```

Data Structures

- struct eap_tls_data

Functions

- int **eap_peer_tls_register** (void)

15.236.1 Detailed Description

EAP peer method: EAP-TLS (RFC 2716).

Copyright

15.237 src/eap_server/eap_tls.c File Reference

hostapd / EAP-TLS (RFC 2716) `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

`#include "eap_tls_common.h"`

`#include "tls.h"`

Data Structures

- struct eap_tls_data

Functions

- int **eap_server_tls_register** (void)

15.237.1 Detailed Description

hostapd / EAP-TLS (RFC 2716)

Copyright

15.238 src/eap_peer/eap_tls_common.c File Reference

EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions. `#include "includes.h"`

```
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "eap_config.h"
#include "sha1.h"
#include "tls.h"
```

Functions

- int eap_peer_tls_ssl_init (struct eap_sm *sm, struct eap_ssl_data *data, struct eap_peer_config *config)

  Initialize shared TLS functionality.

- void eap_peer_tls_ssl_deinit (struct eap_sm *sm, struct eap_ssl_data *data)

  Deinitialize shared TLS functionality.

- u8 * eap_peer_tls_derive_key (struct eap_sm *sm, struct eap_ssl_data *data, const char *label, size_t len)

  Derive a key based on TLS session data.

- const u8 * eap_peer_tls_data_reassemble (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_peer_tls_process_helper (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_type, int peap_version, u8 id, const u8 *in_data, size_t in_len, struct wpabuf **out_data)

  Process TLS handshake message.

- struct wpabuf * eap_peer_tls_build_ack (u8 id, EapType eap_type, int peap_version)

  Build a TLS ACK frame.

- int eap_peer_tls_reauth_init (struct eap_sm *sm, struct eap_ssl_data *data)

  Re-initialize shared TLS for session resumption.

- int eap_peer_tls_status (struct eap_sm *sm, struct eap_ssl_data *data, char *buf, size_t buflen, int verbose)

  Get TLS status.

- const u8 * eap_peer_tls_process_init (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_-type, struct eap_method_ret *ret, const struct wpabuf *reqData, size_t *len, u8 *flags)

  Initial validation/processing of EAP requests.

- void eap_peer_tls_reset_input (struct eap_ssl_data *data)

  Reset input buffers.

- void eap_peer_tls_reset_output (struct eap_ssl_data *data)

  Reset output buffers.

- int eap_peer_tls_decrypt (struct eap_sm *sm, struct eap_ssl_data *data, const struct wpabuf *in_data, struct wpabuf **in_decrypted)

  Decrypt received phase 2 TLS message.

- int eap_peer_tls_encrypt (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_type, int peap_version, u8 id, const struct wpabuf *in_data, struct wpabuf **out_data)

  Encrypt phase 2 TLS message.

- int eap_peer_select_phase2_methods (struct eap_peer_config *config, const char *prefix, struct eap_method_type **types, size_t *num_types)

  Select phase 2 EAP method.

- int eap_peer_tls_phase2_nak (struct eap_method_type *types, size_t num_types, struct eap_hdr *hdr, struct wpabuf **resp)

  Generate EAP-Nak for Phase 2.

15.238.1 Detailed Description

EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions.

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.238.2 Function Documentation

15.238.2.1 int eap_peer_select_phase2_methods (struct eap_peer_config * *config*, const char * *prefix*, struct eap_method_type ** *types*, size_t * *num_types*)

Select phase 2 EAP method.

Parameters:

config Pointer to the network configuration

prefix 'phase2' configuration prefix, e.g., "auth="

types Buffer for returning allocated list of allowed EAP methods

num_types Buffer for returning number of allocated EAP methods

Returns:

0 on success, -1 on failure

This function is used to parse EAP method list and select allowed methods for Phase2 authentication.

15.238.2.2 struct wpabuf* eap_peer_tls_build_ack (u8 *id*, EapType *eap_type*, int *peap_version*) [read]

Build a TLS ACK frame.

Parameters:

id EAP identifier for the response

eap_type EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

peap_version Version number for EAP-PEAP/TTLS

Returns:

Pointer to the allocated ACK frame or NULL on failure

15.238.2.3 const u8* eap_peer_tls_data_reassemble (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:

data Data for TLS processing

in_data Next incoming TLS segment

in_len Length of in_data

out_len Variable for returning length of the reassembled message

need_more_input Variable for returning whether more input data is needed to reassemble this TLS packet

Returns:

Pointer to output data, NULL on error or when more data is needed for the full message (in which case, *need_more_input is also set to 1).

This function reassembles TLS fragments. Caller must not free the returned data buffer since an internal pointer to it is maintained.

15.238.2.4 int eap_peer_tls_decrypt (struct eap_sm * *sm*, struct eap_ssl_data * *data*, const struct wpabuf * *in_data*, struct wpabuf ** *in_decrypted*)

Decrypt received phase 2 TLS message.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

in_data Message received from the server

in_decrypted Buffer for returning a pointer to the decrypted message

Returns:

0 on success, 1 if more input data is needed, or -1 on failure

15.238.2.5 u8* eap_peer_tls_derive_key (struct eap_sm * *sm*, struct eap_ssl_data * *data*, const char * *label*, size_t *len*)

Derive a key based on TLS session data.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

label Label string for deriving the keys, e.g., "client EAP encryption"

len Length of the key material to generate (usually 64 for MSK)

Returns:

Pointer to allocated key on success or NULL on failure

This function uses TLS-PRF to generate pseudo-random data based on the TLS session data (client/server random and master key). Each key type may use a different label to bind the key usage into the generated material.

The caller is responsible for freeing the returned buffer.

15.238.2.6 int eap_peer_tls_encrypt (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, int *peap_version*, u8 *id*, const struct wpabuf * *in_data*, struct wpabuf ** *out_data*)

Encrypt phase 2 TLS message.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

eap_type EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

peap_version Version number for EAP-PEAP/TTLS

id EAP identifier for the response

in_data Plaintext phase 2 data to encrypt or NULL to continue fragments

out_data Buffer for returning a pointer to the encrypted response message

Returns:

0 on success, -1 on failure

15.238.2.7 int eap_peer_tls_phase2_nak (struct eap_method_type * *types*, size_t *num_types*, struct eap_hdr * *hdr*, struct wpabuf ** *resp*)

Generate EAP-Nak for Phase 2.

Parameters:

types Buffer for returning allocated list of allowed EAP methods

num_types Buffer for returning number of allocated EAP methods

hdr EAP-Request header (and the following EAP type octet)

resp Buffer for returning the EAP-Nak message

Returns:

0 on success, -1 on failure

15.238.2.8 int eap_peer_tls_process_helper (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, int *peap_version*, u8 *id*, const u8 * *in_data*, size_t *in_len*, struct wpabuf ** *out_data*)

Process TLS handshake message.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

eap_type EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

peap_version Version number for EAP-PEAP/TTLS

id EAP identifier for the response

in_data Message received from the server

in_len Length of in_data

out_data Buffer for returning a pointer to the response message

Returns:

0 on success, 1 if more input data is needed, 2 if application data is available, or -1 on failure

This function can be used to process TLS handshake messages. It reassembles the received fragments and uses a TLS library to process the messages. The response data from the TLS library is fragmented to suitable output messages that the caller can send out.

out_data is used to return the response message if the return value of this function is 0, 2, or -1. In case of failure, the message is likely a TLS alarm message. The caller is responsible for freeing the allocated buffer if *out_data is not NULL.

This function is called for each received TLS message during the TLS handshake after eap_peer_tls_process_init() call and possible processing of TLS Flags field. Once the handshake has been completed, i.e., when tls_connection_established() returns 1, EAP method specific decrypting of the tunneled data is used.

15.238.2.9 const u8* eap_peer_tls_process_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, struct eap_method_ret * *ret*, const struct wpabuf * *reqData*, size_t * *len*, u8 * *flags*)

Initial validation/processing of EAP requests.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

eap_type EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

ret Return values from EAP request validation and processing

reqData EAP request to be processed (eapReqData)

len Buffer for returning length of the remaining payload

flags Buffer for returning TLS flags

Returns:

Pointer to payload after TLS flags and length or NULL on failure

This function validates the EAP header and processes the optional TLS Message Length field. If this is the first fragment of a TLS message, the TLS reassembly code is initialized to receive the indicated number of bytes.

EAP-TLS, EAP-PEAP, EAP-TTLS, and EAP-FAST methods are expected to use this function as the first step in processing received messages. They will need to process the flags (apart from Message Length Included) that are returned through the flags pointer and the message payload that will be returned (and the length is returned through the len pointer). Return values (ret) are set for continuation of EAP method processing. The caller is responsible for setting these to indicate completion (either success or failure) based on the authentication result.

15.238.2.10 int eap_peer_tls_reauth_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*)

Re-initialize shared TLS for session resumption.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

Returns:

0 on success, -1 on failure

15.238.2.11 void eap_peer_tls_reset_input (struct eap_ssl_data * *data*)

Reset input buffers.

Parameters:

data Data for TLS processing

This function frees any allocated memory for input buffers and resets input state.

15.238.2.12 void eap_peer_tls_reset_output (struct eap_ssl_data * *data*)

Reset output buffers.

Parameters:

data Data for TLS processing

This function frees any allocated memory for output buffers and resets output state.

15.238.2.13 void eap_peer_tls_ssl_deinit (struct eap_sm * *sm*, struct eap_ssl_data * *data*)

Deinitialize shared TLS functionality.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

This function deinitializes shared TLS functionality that was initialized with eap_peer_tls_ssl_init().

15.238.2.14 int eap_peer_tls_ssl_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*, struct eap_peer_config * *config*)

Initialize shared TLS functionality.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

config Pointer to the network configuration

Returns:

0 on success, -1 on failure

This function is used to initialize shared TLS functionality for EAP-TLS, EAP-PEAP, EAP-TTLS, and EAP-FAST.

15.238.2.15 int eap_peer_tls_status (struct eap_sm * *sm*, struct eap_ssl_data * *data*, char * *buf*, size_t *buflen*, int *verbose*)

Get TLS status.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

buf Buffer for status information

buflen Maximum buffer length

verbose Whether to include verbose status information

Returns:

Number of bytes written to buf.

15.239 src/eap_server/eap_tls_common.c File Reference

hostapd / EAP-TLS/PEAP/TTLS/FAST common functions `#include "includes.h"`

```
#include "common.h"
#include "eap_i.h"
#include "eap_tls_common.h"
#include "sha1.h"
#include "tls.h"
```

Functions

- int **eap_server_tls_ssl_init** (struct eap_sm *sm, struct eap_ssl_data *data, int verify_peer)
- void **eap_server_tls_ssl_deinit** (struct eap_sm *sm, struct eap_ssl_data *data)
- u8 * **eap_server_tls_derive_key** (struct eap_sm *sm, struct eap_ssl_data *data, char *label, size_t len)
- struct wpabuf * **eap_server_tls_build_msg** (struct eap_ssl_data *data, int eap_type, int version, u8 id)
- struct wpabuf * **eap_server_tls_build_ack** (u8 id, int eap_type, int version)
- int **eap_server_tls_phase1** (struct eap_sm *sm, struct eap_ssl_data *data)
- struct wpabuf * **eap_server_tls_encrypt** (struct eap_sm *sm, struct eap_ssl_data *data, const u8 *plain, size_t plain_len)
- int **eap_server_tls_process** (struct eap_sm *sm, struct eap_ssl_data *data, struct wpabuf *respData, void *priv, int eap_type, int(*proc_version)(struct eap_sm *sm, void *priv, int peer_version), void(*proc_msg)(struct eap_sm *sm, void *priv, const struct wpabuf *respData))

15.239.1 Detailed Description

hostapd / EAP-TLS/PEAP/TTLS/FAST common functions

Copyright

15.240 src/eap_peer/eap_tls_common.h File Reference

EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions.

Data Structures

- struct eap_ssl_data

  TLS data for EAP methods.

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_peer_tls_ssl_init (struct eap_sm *sm, struct eap_ssl_data *data, struct eap_peer_config *config)

  Initialize shared TLS functionality.

- void eap_peer_tls_ssl_deinit (struct eap_sm *sm, struct eap_ssl_data *data)

  Deinitialize shared TLS functionality.

- u8 * eap_peer_tls_derive_key (struct eap_sm *sm, struct eap_ssl_data *data, const char *label, size_t len)

  Derive a key based on TLS session data.

- const u8 * eap_peer_tls_data_reassemble (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_peer_tls_process_helper (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_type, int peap_version, u8 id, const u8 *in_data, size_t in_len, struct wpabuf **out_data)

  Process TLS handshake message.

- struct wpabuf * eap_peer_tls_build_ack (u8 id, EapType eap_type, int peap_version)

  Build a TLS ACK frame.

- int eap_peer_tls_reauth_init (struct eap_sm *sm, struct eap_ssl_data *data)

  Re-initialize shared TLS for session resumption.

- int eap_peer_tls_status (struct eap_sm *sm, struct eap_ssl_data *data, char *buf, size_t buflen, int verbose)

  Get TLS status.

- const u8 * eap_peer_tls_process_init (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_-type, struct eap_method_ret *ret, const struct wpabuf *reqData, size_t *len, u8 *flags)

  Initial validation/processing of EAP requests.

- void eap_peer_tls_reset_input (struct eap_ssl_data *data)

  Reset input buffers.

- void eap_peer_tls_reset_output (struct eap_ssl_data *data)

  Reset output buffers.

- int eap_peer_tls_decrypt (struct eap_sm *sm, struct eap_ssl_data *data, const struct wpabuf *in_-data, struct wpabuf **in_decrypted)

  Decrypt received phase 2 TLS message.

- int eap_peer_tls_encrypt (struct eap_sm *sm, struct eap_ssl_data *data, EapType eap_type, int peap_version, u8 id, const struct wpabuf *in_data, struct wpabuf **out_data)

  Encrypt phase 2 TLS message.

- int eap_peer_select_phase2_methods (struct eap_peer_config *config, const char *prefix, struct eap_method_type **types, size_t *num_types)

  Select phase 2 EAP method.

- int eap_peer_tls_phase2_nak (struct eap_method_type *types, size_t num_types, struct eap_hdr *hdr, struct wpabuf **resp)

  Generate EAP-Nak for Phase 2.

15.240.1 Detailed Description

EAP peer: EAP-TLS/PEAP/TTLS/FAST common functions.

Copyright

Copyright (c) 2004-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.240.2 Function Documentation

15.240.2.1 int eap_peer_select_phase2_methods (struct eap_peer_config * *config*, const char * *prefix*, struct eap_method_type ** *types*, size_t * *num_types*)

Select phase 2 EAP method.

Parameters:

config Pointer to the network configuration

prefix 'phase2' configuration prefix, e.g., "auth="

types Buffer for returning allocated list of allowed EAP methods

num_types Buffer for returning number of allocated EAP methods

Returns:

0 on success, -1 on failure

This function is used to parse EAP method list and select allowed methods for Phase2 authentication.

15.240.2.2 struct wpabuf* eap_peer_tls_build_ack (u8 *id*, EapType *eap_type*, int *peap_version*) [read]

Build a TLS ACK frame.

Parameters:

id EAP identifier for the response

eap_type EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

peap_version Version number for EAP-PEAP/TTLS

Returns:

Pointer to the allocated ACK frame or NULL on failure

15.240.2.3 const u8* eap_peer_tls_data_reassemble (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:

data Data for TLS processing

in_data Next incoming TLS segment

in_len Length of in_data

out_len Variable for returning length of the reassembled message

need_more_input Variable for returning whether more input data is needed to reassemble this TLS packet

Returns:

Pointer to output data, NULL on error or when more data is needed for the full message (in which case, *need_more_input is also set to 1).

This function reassembles TLS fragments. Caller must not free the returned data buffer since an internal pointer to it is maintained.

15.240.2.4 int eap_peer_tls_decrypt (struct eap_sm * *sm*, struct eap_ssl_data * *data*, const struct wpabuf * *in_data*, struct wpabuf ** *in_decrypted*)

Decrypt received phase 2 TLS message.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

in_data Message received from the server

in_decrypted Buffer for returning a pointer to the decrypted message

Returns:

0 on success, 1 if more input data is needed, or -1 on failure

15.240.2.5 u8* eap_peer_tls_derive_key (struct eap_sm * *sm*, struct eap_ssl_data * *data*, const char * *label*, size_t *len*)

Derive a key based on TLS session data.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

label Label string for deriving the keys, e.g., "client EAP encryption"

len Length of the key material to generate (usually 64 for MSK)

Returns:

Pointer to allocated key on success or NULL on failure

This function uses TLS-PRF to generate pseudo-random data based on the TLS session data (client/server random and master key). Each key type may use a different label to bind the key usage into the generated material.

The caller is responsible for freeing the returned buffer.

15.240.2.6 int eap_peer_tls_encrypt (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, int *peap_version*, u8 *id*, const struct wpabuf * *in_data*, struct wpabuf ** *out_data*)

Encrypt phase 2 TLS message.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

eap_type EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

peap_version Version number for EAP-PEAP/TTLS

id EAP identifier for the response

in_data Plaintext phase 2 data to encrypt or NULL to continue fragments

out_data Buffer for returning a pointer to the encrypted response message

Returns:

0 on success, -1 on failure

15.240.2.7 int eap_peer_tls_phase2_nak (struct eap_method_type * *types*, size_t *num_types*, struct eap_hdr * *hdr*, struct wpabuf ** *resp*)

Generate EAP-Nak for Phase 2.

Parameters:

types Buffer for returning allocated list of allowed EAP methods

num_types Buffer for returning number of allocated EAP methods

hdr EAP-Request header (and the following EAP type octet)

resp Buffer for returning the EAP-Nak message

Returns:

0 on success, -1 on failure

15.240.2.8 int eap_peer_tls_process_helper (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, int *peap_version*, u8 *id*, const u8 * *in_data*, size_t *in_len*, struct wpabuf ** *out_data*)

Process TLS handshake message.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

eap_type EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

peap_version Version number for EAP-PEAP/TTLS

id EAP identifier for the response

in_data Message received from the server

in_len Length of in_data

out_data Buffer for returning a pointer to the response message

Returns:

0 on success, 1 if more input data is needed, 2 if application data is available, or -1 on failure

This function can be used to process TLS handshake messages. It reassembles the received fragments and uses a TLS library to process the messages. The response data from the TLS library is fragmented to suitable output messages that the caller can send out.

out_data is used to return the response message if the return value of this function is 0, 2, or -1. In case of failure, the message is likely a TLS alarm message. The caller is responsible for freeing the allocated buffer if *out_data is not NULL.

This function is called for each received TLS message during the TLS handshake after eap_peer_tls_process_init() call and possible processing of TLS Flags field. Once the handshake has been completed, i.e., when tls_connection_established() returns 1, EAP method specific decrypting of the tunneled data is used.

15.240.2.9 const u8* eap_peer_tls_process_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*, EapType *eap_type*, struct eap_method_ret * *ret*, const struct wpabuf * *reqData*, size_t * *len*, u8 * *flags*)

Initial validation/processing of EAP requests.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

eap_type EAP type (EAP_TYPE_TLS, EAP_TYPE_PEAP, ...)

ret Return values from EAP request validation and processing

reqData EAP request to be processed (eapReqData)

len Buffer for returning length of the remaining payload

flags Buffer for returning TLS flags

Returns:

Pointer to payload after TLS flags and length or NULL on failure

This function validates the EAP header and processes the optional TLS Message Length field. If this is the first fragment of a TLS message, the TLS reassembly code is initialized to receive the indicated number of bytes.

EAP-TLS, EAP-PEAP, EAP-TTLS, and EAP-FAST methods are expected to use this function as the first step in processing received messages. They will need to process the flags (apart from Message Length Included) that are returned through the flags pointer and the message payload that will be returned (and the length is returned through the len pointer). Return values (ret) are set for continuation of EAP method processing. The caller is responsible for setting these to indicate completion (either success or failure) based on the authentication result.

15.240.2.10 int eap_peer_tls_reauth_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*)

Re-initialize shared TLS for session resumption.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

Returns:

0 on success, -1 on failure

15.240.2.11 void eap_peer_tls_reset_input (struct eap_ssl_data * *data*)

Reset input buffers.

Parameters:

data Data for TLS processing

This function frees any allocated memory for input buffers and resets input state.

15.240.2.12 void eap_peer_tls_reset_output (struct eap_ssl_data * *data*)

Reset output buffers.

Parameters:

data Data for TLS processing

This function frees any allocated memory for output buffers and resets output state.

15.240.2.13 void eap_peer_tls_ssl_deinit (struct eap_sm * *sm*, struct eap_ssl_data * *data*)

Deinitialize shared TLS functionality.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

This function deinitializes shared TLS functionality that was initialized with eap_peer_tls_ssl_init().

15.240.2.14 int eap_peer_tls_ssl_init (struct eap_sm * *sm*, struct eap_ssl_data * *data*, struct eap_peer_config * *config*)

Initialize shared TLS functionality.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

config Pointer to the network configuration

Returns:

0 on success, -1 on failure

This function is used to initialize shared TLS functionality for EAP-TLS, EAP-PEAP, EAP-TTLS, and EAP-FAST.

15.240.2.15 int eap_peer_tls_status (struct eap_sm * *sm*, struct eap_ssl_data * *data*, char * *buf*, size_t *buflen*, int *verbose*)

Get TLS status.

Parameters:

sm Pointer to EAP state machine allocated with eap_peer_sm_init()

data Data for TLS processing

buf Buffer for status information

buflen Maximum buffer length

verbose Whether to include verbose status information

Returns:

Number of bytes written to buf.

15.241 src/eap_server/eap_tls_common.h File Reference

hostapd / EAP-TLS/PEAP/TTLS/FAST common functions

Data Structures

- struct eap_ssl_data

  TLS data for EAP methods.

Defines

- #define **EAP_TLS_FLAGS_LENGTH_INCLUDED** 0x80
- #define **EAP_TLS_FLAGS_MORE_FRAGMENTS** 0x40
- #define **EAP_TLS_FLAGS_START** 0x20
- #define **EAP_TLS_VERSION_MASK** 0x07
- #define **EAP_TLS_KEY_LEN** 64

Functions

- int **eap_server_tls_ssl_init** (struct eap_sm *sm, struct eap_ssl_data *data, int verify_peer)
- void **eap_server_tls_ssl_deinit** (struct eap_sm *sm, struct eap_ssl_data *data)
- u8 * **eap_server_tls_derive_key** (struct eap_sm *sm, struct eap_ssl_data *data, char *label, size_t len)
- struct wpabuf * **eap_server_tls_build_msg** (struct eap_ssl_data *data, int eap_type, int version, u8 id)
- struct wpabuf * **eap_server_tls_build_ack** (u8 id, int eap_type, int version)
- int **eap_server_tls_phase1** (struct eap_sm *sm, struct eap_ssl_data *data)
- struct wpabuf * **eap_server_tls_encrypt** (struct eap_sm *sm, struct eap_ssl_data *data, const u8 *plain, size_t plain_len)
- int **eap_server_tls_process** (struct eap_sm *sm, struct eap_ssl_data *data, struct wpabuf *respData, void *priv, int eap_type, int(*proc_version)(struct eap_sm *sm, void *priv, int peer_version), void(*proc_msg)(struct eap_sm *sm, void *priv, const struct wpabuf *respData))

15.241.1 Detailed Description

hostapd / EAP-TLS/PEAP/TTLS/FAST common functions

Copyright

15.242 src/eap_peer/eap_tnc.c File Reference

EAP peer method: EAP-TNC (Trusted Network Connect). `#include "includes.h"`

`#include "common.h"`

`#include "base64.h"`

`#include "eap_i.h"`

`#include "tncc.h"`

Data Structures

- struct eap_tnc_data

Defines

- #define **EAP_TNC_FLAGS_LENGTH_INCLUDED** 0x80
- #define **EAP_TNC_FLAGS_MORE_FRAGMENTS** 0x40
- #define **EAP_TNC_FLAGS_START** 0x20
- #define **EAP_TNC_VERSION_MASK** 0x07
- #define **EAP_TNC_VERSION** 1

Functions

- int **eap_peer_tnc_register** (void)

15.242.1 Detailed Description

EAP peer method: EAP-TNC (Trusted Network Connect).

Copyright

15.243 src/eap_server/eap_tnc.c File Reference

EAP server method: EAP-TNC (Trusted Network Connect). `#include "includes.h"`

```
#include "common.h"
#include "base64.h"
#include "eap_i.h"
#include "tncs.h"
```

Data Structures

- struct eap_tnc_data

Defines

- #define **EAP_TNC_FLAGS_LENGTH_INCLUDED** 0x80
- #define **EAP_TNC_FLAGS_MORE_FRAGMENTS** 0x40
- #define **EAP_TNC_FLAGS_START** 0x20
- #define **EAP_TNC_VERSION_MASK** 0x07
- #define **EAP_TNC_VERSION** 1

Functions

- int **eap_server_tnc_register** (void)

15.243.1 Detailed Description

EAP server method: EAP-TNC (Trusted Network Connect).

Copyright

15.244 src/eap_peer/eap_ttls.c File Reference

EAP peer method: EAP-TTLS (RFC 5281). `#include "includes.h"`

```
#include "common.h"
#include "eap_peer/eap_i.h"
#include "eap_peer/eap_tls_common.h"
#include "eap_peer/eap_config.h"
#include "ms_funcs.h"
#include "sha1.h"
#include "eap_common/chap.h"
#include "tls.h"
#include "mschapv2.h"
#include "eap_common/eap_ttls.h"
```

Data Structures

- struct eap_ttls_data
- struct ttls_parse_avp

Defines

- #define **EAP_TTLS_VERSION** 0
- #define **MSCHAPV2_KEY_LEN** 16
- #define **MSCHAPV2_NT_RESPONSE_LEN** 24

Functions

- int **eap_peer_ttls_register** (void)

15.244.1 Detailed Description

EAP peer method: EAP-TTLS (RFC 5281).

Copyright

15.245 src/eap_server/eap_ttls.c File Reference

hostapd / EAP-TTLS (RFC 5281) `#include "includes.h"`

```
#include "common.h"
#include "eap_server/eap_i.h"
#include "eap_server/eap_tls_common.h"
#include "ms_funcs.h"
#include "sha1.h"
#include "eap_common/chap.h"
#include "tls.h"
#include "eap_common/eap_ttls.h"
```

Data Structures

- struct eap_ttls_data
- struct eap_ttls_avp

Defines

- #define **MSCHAPV2_KEY_LEN** 16

Functions

- int **eap_server_ttls_register** (void)

15.245.1 Detailed Description

hostapd / EAP-TTLS (RFC 5281)

Copyright

15.246 src/eap_peer/eap_vendor_test.c File Reference

EAP peer method: Test method for vendor specific (expanded) EAP type. `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

Data Structures

- struct eap_vendor_test_data

Defines

- #define **EAP_VENDOR_ID** 0xfffefd
- #define **EAP_VENDOR_TYPE** 0xfcfbfaf9

Functions

- int **eap_peer_vendor_test_register** (void)

15.246.1 Detailed Description

EAP peer method: Test method for vendor specific (expanded) EAP type.

Copyright

This file implements a vendor specific test method using EAP expanded types. This is only for test use and must not be used for authentication since no security is provided.

15.247 src/eap_server/eap_vendor_test.c File Reference

hostapd / Test method for vendor specific (expanded) EAP type `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

Data Structures

- struct eap_vendor_test_data

Defines

- #define **EAP_VENDOR_ID** 0xfffefd
- #define **EAP_VENDOR_TYPE** 0xfcfbfaf9

Functions

- int **eap_server_vendor_test_register** (void)

15.247.1 Detailed Description

hostapd / Test method for vendor specific (expanded) EAP type

Copyright

15.248 src/eap_peer/eap_wsc.c File Reference

EAP-WSC peer for Wi-Fi Protected Setup. `#include "includes.h"`

```
#include "common.h"
#include "uuid.h"
#include "eap_i.h"
#include "eap_common/eap_wsc_common.h"
#include "wps/wps.h"
#include "wps/wps_defs.h"
```

Data Structures

- struct eap_wsc_data

Functions

- int **eap_peer_wsc_register** (void)

15.248.1 Detailed Description

EAP-WSC peer for Wi-Fi Protected Setup.

Copyright

15.249 src/eap_server/eap_wsc.c File Reference

EAP-WSC server for Wi-Fi Protected Setup. `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "eap_i.h"
#include "eap_common/eap_wsc_common.h"
#include "wps/wps.h"
```

Data Structures

- struct eap_wsc_data

Functions

- int **eap_server_wsc_register** (void)

15.249.1 Detailed Description

EAP-WSC server for Wi-Fi Protected Setup.

Copyright

15.250 src/eap_peer/ikev2.c File Reference

IKEv2 responder (RFC 4306) for EAP-IKEV2. `#include "includes.h"`

`#include "common.h"`

`#include "dh_groups.h"`

`#include "ikev2.h"`

Functions

- void **ikev2_responder_deinit** (struct ikev2_responder_data *data)
- int **ikev2_responder_process** (struct ikev2_responder_data *data, const struct wpabuf *buf)
- struct wpabuf * **ikev2_responder_build** (struct ikev2_responder_data *data)

15.250.1 Detailed Description

IKEv2 responder (RFC 4306) for EAP-IKEV2.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.251 src/eap_server/ikev2.c File Reference

IKEv2 initiator (RFC 4306) for EAP-IKEV2. `#include "includes.h"`

`#include "common.h"`

`#include "dh_groups.h"`

`#include "ikev2.h"`

Functions

- void **ikev2_initiator_deinit** (struct ikev2_initiator_data *data)
- int **ikev2_initiator_process** (struct ikev2_initiator_data *data, const struct wpabuf *buf)
- struct wpabuf * **ikev2_initiator_build** (struct ikev2_initiator_data *data)

15.251.1 Detailed Description

IKEv2 initiator (RFC 4306) for EAP-IKEV2.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.252 src/eap_peer/ikev2.h File Reference

IKEv2 responder (RFC 4306) for EAP-IKEV2. `#include "eap_common/ikev2_common.h"`

Data Structures

- struct ikev2_proposal_data
- struct ikev2_responder_data

Functions

- void **ikev2_responder_deinit** (struct ikev2_responder_data *data)
- int **ikev2_responder_process** (struct ikev2_responder_data *data, const struct wpabuf *buf)
- struct wpabuf * **ikev2_responder_build** (struct ikev2_responder_data *data)

15.252.1 Detailed Description

IKEv2 responder (RFC 4306) for EAP-IKEV2.

Copyright

15.253 src/eap_server/ikev2.h File Reference

IKEv2 initiator (RFC 4306) for EAP-IKEV2. `#include "eap_common/ikev2_common.h"`

Data Structures

- struct ikev2_proposal_data
- struct ikev2_initiator_data

Functions

- void **ikev2_initiator_deinit** (struct ikev2_initiator_data *data)
- int **ikev2_initiator_process** (struct ikev2_initiator_data *data, const struct wpabuf *buf)
- struct wpabuf * **ikev2_initiator_build** (struct ikev2_initiator_data *data)

15.253.1 Detailed Description

IKEv2 initiator (RFC 4306) for EAP-IKEV2.

Copyright

15.254 src/eap_peer/mschapv2.c File Reference

MSCHAPV2 (RFC 2759). `#include "includes.h"`

`#include "common.h"`

`#include "ms_funcs.h"`

`#include "mschapv2.h"`

Functions

- const u8 * **mschapv2_remove_domain** (const u8 *username, size_t *len)
- int **mschapv2_derive_response** (const u8 *identity, size_t identity_len, const u8 *password, size_t password_len, int pwhash, const u8 *auth_challenge, const u8 *peer_challenge, u8 *nt_response, u8 *auth_response, u8 *master_key)
- int **mschapv2_verify_auth_response** (const u8 *auth_response, const u8 *buf, size_t buf_len)

15.254.1 Detailed Description

MSCHAPV2 (RFC 2759).

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.255 src/eap_peer/mschapv2.h File Reference

MSCHAPV2 (RFC 2759).

Defines

- #define **MSCHAPV2_CHAL_LEN** 16
- #define **MSCHAPV2_NT_RESPONSE_LEN** 24
- #define **MSCHAPV2_AUTH_RESPONSE_LEN** 20
- #define **MSCHAPV2_MASTER_KEY_LEN** 16

Functions

- const u8 ∗ **mschapv2_remove_domain** (const u8 ∗username, size_t ∗len)
- int **mschapv2_derive_response** (const u8 ∗username, size_t username_len, const u8 ∗password, size_t password_len, int pwhash, const u8 ∗auth_challenge, const u8 ∗peer_challenge, u8 ∗nt_response, u8 ∗auth_response, u8 ∗master_key)
- int **mschapv2_verify_auth_response** (const u8 ∗auth_response, const u8 ∗buf, size_t buf_len)

15.255.1 Detailed Description

MSCHAPV2 (RFC 2759).

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.256 src/eap_peer/tncc.c File Reference

EAP-TNC - TNCC (IF-IMC and IF-TNCCS). `#include "includes.h"`

```
#include <dlfcn.h>
#include "common.h"
#include "base64.h"
#include "tncc.h"
#include "eap_common/eap_tlv_common.h"
#include "eap_common/eap_defs.h"
```

Data Structures

- struct tnc_if_imc
- struct tncc_data

Defines

- #define **TSTR** "%s"
- #define **TNC_CONFIG_FILE** "/etc/tnc_config"
- #define **TNC_WINREG_PATH** TEXT("SOFTWARE\\Trusted Computing Group\\TNC\\IMCs")
- #define **IF_TNCCS_START**
- #define **IF_TNCCS_END** "\n</TNCCS-Batch>"
- #define **TNC_RESULT_SUCCESS** 0
- #define **TNC_RESULT_NOT_INITIALIZED** 1
- #define **TNC_RESULT_ALREADY_INITIALIZED** 2
- #define **TNC_RESULT_NO_COMMON_VERSION** 3
- #define **TNC_RESULT_CANT_RETRY** 4
- #define **TNC_RESULT_WONT_RETRY** 5
- #define **TNC_RESULT_INVALID_PARAMETER** 6
- #define **TNC_RESULT_CANT_RESPOND** 7
- #define **TNC_RESULT_ILLEGAL_OPERATION** 8
- #define **TNC_RESULT_OTHER** 9
- #define **TNC_RESULT_FATAL** 10
- #define **TNC_CONNECTION_STATE_CREATE** 0
- #define **TNC_CONNECTION_STATE_HANDSHAKE** 1
- #define **TNC_CONNECTION_STATE_ACCESS_ALLOWED** 2
- #define **TNC_CONNECTION_STATE_ACCESS_ISOLATED** 3
- #define **TNC_CONNECTION_STATE_ACCESS_NONE** 4
- #define **TNC_CONNECTION_STATE_DELETE** 5
- #define **TNC_IFIMC_VERSION_1** 1
- #define **TNC_VENDORID_ANY** ((TNC_VendorID) 0xffffff)
- #define **TNC_SUBTYPE_ANY** ((TNC_MessageSubtype) 0xff)
- #define **TNC_TNCCS_RECOMMENDATION** 0x00000001
- #define **TNC_TNCCS_ERROR** 0x00000002
- #define **TNC_TNCCS_PREFERREDLANGUAGE** 0x00000003
- #define **TNC_TNCCS_REASONSTRINGS** 0x00000004
- #define **TNC_MAX_IMC_ID** 10

Typedefs

- typedef unsigned long **TNC_UInt32**
- typedef unsigned char * **TNC_BufferReference**
- typedef TNC_UInt32 **TNC_IMCID**
- typedef TNC_UInt32 **TNC_ConnectionID**
- typedef TNC_UInt32 **TNC_ConnectionState**
- typedef TNC_UInt32 **TNC_RetryReason**
- typedef TNC_UInt32 **TNC_MessageType**
- typedef TNC_MessageType * **TNC_MessageTypeList**
- typedef TNC_UInt32 **TNC_VendorID**
- typedef TNC_UInt32 **TNC_MessageSubtype**
- typedef TNC_UInt32 **TNC_Version**
- typedef TNC_UInt32 **TNC_Result**
- typedef TNC_Result(* **TNC_TNCC_BindFunctionPointer**)(TNC_IMCID imcID, char *functionName, void **pOutfunctionPointer)

Enumerations

- enum {

  SSOH_MS_MACHINE_INVENTORY = 1, **SSOH_MS_QUARANTINE_STATE** = 2, **SSOH_MS_PACKET_INFO** = 3, **SSOH_MS_SYSTEMGENERATED_IDS** = 4,

  SSOH_MS_MACHINENAME = 5, **SSOH_MS_CORRELATIONID** = 6, **SSOH_MS_INSTALLED_SHVS** = 7, **SSOH_MS_MACHINE_INVENTORY_EX** = 8 }

Functions

- TNC_Result **TNC_TNCC_ReportMessageTypes** (TNC_IMCID imcID, TNC_MessageTypeList supportedTypes, TNC_UInt32 typeCount)
- TNC_Result **TNC_TNCC_SendMessage** (TNC_IMCID imcID, TNC_ConnectionID connectionID, TNC_BufferReference message, TNC_UInt32 messageLength, TNC_MessageType messageType)
- TNC_Result **TNC_TNCC_RequestHandshakeRetry** (TNC_IMCID imcID, TNC_ConnectionID connectionID, TNC_RetryReason reason)
- TNC_Result **TNC_9048_LogMessage** (TNC_IMCID imcID, TNC_UInt32 severity, const char *message)
- TNC_Result **TNC_9048_UserMessage** (TNC_IMCID imcID, TNC_ConnectionID connectionID, const char *message)
- TNC_Result **TNC_TNCC_BindFunction** (TNC_IMCID imcID, char *functionName, void **pOutfunctionPointer)
- void **tncc_init_connection** (struct tncc_data *tncc)
- size_t **tncc_total_send_len** (struct tncc_data *tncc)
- u8 * **tncc_copy_send_buf** (struct tncc_data *tncc, u8 *pos)
- char * **tncc_if_tnccs_start** (struct tncc_data *tncc)
- char * **tncc_if_tnccs_end** (void)
- enum tncc_process_res **tncc_process_if_tnccs** (struct tncc_data *tncc, const u8 *msg, size_t len)
- struct tncc_data * **tncc_init** (void)
- void **tncc_deinit** (struct tncc_data *tncc)
- struct wpabuf * **tncc_process_soh_request** (int ver, const u8 *data, size_t len)

15.256.1 Detailed Description

EAP-TNC - TNCC (IF-IMC and IF-TNCCS).

Copyright

15.256.2 Define Documentation

15.256.2.1 #define IF_TNCCS_START

Value:

```
"<?xml version=\"1.0\"?>\n" \
"<TNCCS-Batch BatchId=\"%d\" Recipient=\"TNCS\" " \
"xmlns=\"http://www.trustedcomputinggroup.org/IWG/TNC/1_0/IF_TNCCS#\" " \
"xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " \
"xsi:schemaLocation=\"http://www.trustedcomputinggroup.org/IWG/TNC/1_0/" \
"IF_TNCCS# https://www.trustedcomputinggroup.org/XML/SCHEMA/TNCCS_1.0.xsd\">\n"
```

15.257 src/eap_peer/tncc.h File Reference

EAP-TNC - TNCC (IF-IMC and IF-TNCCS).

Enumerations

- enum **tncc_process_res** {
  TNCCS_PROCESS_ERROR = -1, **TNCCS_PROCESS_OK_NO_RECOMMENDATION** = 0, **TNCCS_RECOMMENDATION_ERROR**, **TNCCS_RECOMMENDATION_ALLOW**, **TNCCS_RECOMMENDATION_NONE**, **TNCCS_RECOMMENDATION_ISOLATE** }

Functions

- struct tncc_data * **tncc_init** (void)
- void **tncc_deinit** (struct tncc_data *tncc)
- void **tncc_init_connection** (struct tncc_data *tncc)
- size_t **tncc_total_send_len** (struct tncc_data *tncc)
- u8 * **tncc_copy_send_buf** (struct tncc_data *tncc, u8 *pos)
- char * **tncc_if_tnccs_start** (struct tncc_data *tncc)
- char * **tncc_if_tnccs_end** (void)
- enum tncc_process_res **tncc_process_if_tnccs** (struct tncc_data *tncc, const u8 *msg, size_t len)
- struct wpabuf * **tncc_process_soh_request** (int ver, const u8 *data, size_t len)

15.257.1 Detailed Description

EAP-TNC - TNCC (IF-IMC and IF-TNCCS).

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.258 src/eap_server/eap_identity.c File Reference

hostapd / EAP-Identity `#include "includes.h"`

`#include "common.h"`

`#include "eap_i.h"`

Data Structures

- struct eap_identity_data

Functions

- int **eap_server_identity_register** (void)

15.258.1 Detailed Description

hostapd / EAP-Identity

Copyright

Copyright (c) 2004-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.259 src/eap_server/eap_sim_db.c File Reference

hostapd / EAP-SIM database/authenticator gateway `#include "includes.h"`

```
#include <sys/un.h>
#include "common.h"
#include "eap_common/eap_sim_common.h"
#include "eap_server/eap_sim_db.h"
#include "eloop.h"
```

Data Structures

- struct eap_sim_pseudonym
- struct eap_sim_db_pending
- struct eap_sim_db_data

Functions

- void * eap_sim_db_init (const char *config, void(*get_complete_cb)(void *ctx, void *session_ctx), void *ctx)

  Initialize EAP-SIM DB / authentication gateway interface.

- void eap_sim_db_deinit (void *priv)

  Deinitialize EAP-SIM DB/authentication gw interface.

- int eap_sim_db_get_gsm_triplets (void *priv, const u8 *identity, size_t identity_len, int max_chal, u8 *_rand, u8 *kc, u8 *sres, void *cb_session_ctx)

  Get GSM triplets.

- int eap_sim_db_identity_known (void *priv, const u8 *identity, size_t identity_len)

  Verify whether the given identity is known.

- char * eap_sim_db_get_next_pseudonym (void *priv, int aka)

  EAP-SIM DB: Get next pseudonym.

- char * eap_sim_db_get_next_reauth_id (void *priv, int aka)

  EAP-SIM DB: Get next reauth_id.

- int eap_sim_db_add_pseudonym (void *priv, const u8 *identity, size_t identity_len, char *pseudonym)

  EAP-SIM DB: Add new pseudonym.

- int eap_sim_db_add_reauth (void *priv, const u8 *identity, size_t identity_len, char *reauth_id, u16 counter, const u8 *mk)

  EAP-SIM DB: Add new re-authentication entry.

- const u8 * eap_sim_db_get_permanent (void *priv, const u8 *identity, size_t identity_len, size_t *len)

EAP-SIM DB: Get permanent identity.

- struct eap_sim_reauth * eap_sim_db_get_reauth_entry (void *priv, const u8 *identity, size_t identity_len)

  EAP-SIM DB: Get re-authentication entry.

- void eap_sim_db_remove_reauth (void *priv, struct eap_sim_reauth *reauth)

  EAP-SIM DB: Remove re-authentication entry.

- int eap_sim_db_get_aka_auth (void *priv, const u8 *identity, size_t identity_len, u8 *_rand, u8 *autn, u8 *ik, u8 *ck, u8 *res, size_t *res_len, void *cb_session_ctx)

  Get AKA authentication values.

- int eap_sim_db_resynchronize (void *priv, const u8 *identity, size_t identity_len, const u8 *auts, const u8 *_rand)

  Resynchronize AKA AUTN.

15.259.1 Detailed Description

hostapd / EAP-SIM database/authenticator gateway

Copyright

This is an example implementation of the EAP-SIM/AKA database/authentication gateway interface that is using an external program as an SS7 gateway to GSM/UMTS authentication center (HLR/AuC). hlr_auc_gw is an example implementation of such a gateway program. This eap_sim_db.c takes care of EAP-SIM/AKA pseudonyms and re-auth identities. It can be used with different gateway implementations for HLR/AuC access. Alternatively, it can also be completely replaced if the in-memory database of pseudonyms/re-auth identities is not suitable for some cases.

15.259.2 Function Documentation

15.259.2.1 int eap_sim_db_add_pseudonym (void * *priv*, const u8 * *identity*, size_t *identity_len*, char * *pseudonym*)

EAP-SIM DB: Add new pseudonym.

Parameters:

priv Private data pointer from eap_sim_db_init()

identity Identity of the user (may be permanent identity or pseudonym)

identity_len Length of identity

pseudonym Pseudonym for this user. This needs to be an allocated buffer, e.g., return value from eap_sim_db_get_next_pseudonym(). Caller must not free it.

Returns:

0 on success, -1 on failure

This function adds a new pseudonym for EAP-SIM user. EAP-SIM DB is responsible of freeing pseudonym buffer once it is not needed anymore.

15.259.2.2 int eap_sim_db_add_reauth (void * *priv*, const u8 * *identity*, size_t *identity_len*, char * *reauth_id*, u16 *counter*, const u8 * *mk*)

EAP-SIM DB: Add new re-authentication entry.

Parameters:

priv Private data pointer from eap_sim_db_init()

identity Identity of the user (may be permanent identity or pseudonym)

identity_len Length of identity

reauth_id reauth_id for this user. This needs to be an allocated buffer, e.g., return value from eap_sim_db_get_next_reauth_id(). Caller must not free it.

counter AT_COUNTER value for fast re-authentication

mk 16-byte MK from the previous full authentication or NULL

Returns:

0 on success, -1 on failure

This function adds a new re-authentication entry for an EAP-SIM user. EAP-SIM DB is responsible of freeing reauth_id buffer once it is not needed anymore.

15.259.2.3 void eap_sim_db_deinit (void * *priv*)

Deinitialize EAP-SIM DB/authentication gw interface.

Parameters:

priv Private data pointer from eap_sim_db_init()

15.259.2.4 int eap_sim_db_get_aka_auth (void * *priv*, const u8 * *identity*, size_t *identity_len*, u8 * *_rand*, u8 * *autn*, u8 * *ik*, u8 * *ck*, u8 * *res*, size_t * *res_len*, void * *cb_session_ctx*)

Get AKA authentication values.

Parameters:

priv Private data pointer from eap_sim_db_init()

identity User name identity

identity_len Length of identity in bytes

_rand Buffer for RAND value

autn Buffer for AUTN value

ik Buffer for IK value

ck Buffer for CK value

res Buffer for RES value

res_len Buffer for RES length

cb_session_ctx Session callback context for get_complete_cb()

Returns:

0 on success, -1 (EAP_SIM_DB_FAILURE) on error (e.g., user not found), or -2 (EAP_SIM_DB_-PENDING) if results are not yet available. In this case, the callback function registered with eap_-sim_db_init() will be called once the results become available.

In most cases, the user name is '0' | IMSI, i.e., 0 followed by the IMSI in ASCII format.

When using an external server for AKA authentication, this function can always start a request and return EAP_SIM_DB_PENDING immediately if authentication triplets are not available. Once the authentication data are received, callback function registered with eap_sim_db_init() is called to notify EAP state machine to reprocess the message. This eap_sim_db_get_aka_auth() function will then be called again and the newly received triplets will then be given to the caller.

15.259.2.5 int eap_sim_db_get_gsm_triplets (void * *priv*, const u8 * *identity*, size_t *identity_len*, int *max_chal*, u8 * *_rand*, u8 * *kc*, u8 * *sres*, void * *cb_session_ctx*)

Get GSM triplets.

Parameters:

priv Private data pointer from eap_sim_db_init()

identity User name identity

identity_len Length of identity in bytes

max_chal Maximum number of triplets

_rand Buffer for RAND values

kc Buffer for Kc values

sres Buffer for SRES values

cb_session_ctx Session callback context for get_complete_cb()

Returns:

Number of triplets received (has to be less than or equal to max_chal), -1 (EAP_SIM_DB_FAILURE) on error (e.g., user not found), or -2 (EAP_SIM_DB_PENDING) if results are not yet available. In this case, the callback function registered with eap_sim_db_init() will be called once the results become available.

In most cases, the user name is '1' | IMSI, i.e., 1 followed by the IMSI in ASCII format.

When using an external server for GSM triplets, this function can always start a request and return EAP_-SIM_DB_PENDING immediately if authentication triplets are not available. Once the triplets are received, callback function registered with eap_sim_db_init() is called to notify EAP state machine to reprocess the message. This eap_sim_db_get_gsm_triplets() function will then be called again and the newly received triplets will then be given to the caller.

15.259.2.6 char* eap_sim_db_get_next_pseudonym (void * *priv*, int *aka*)

EAP-SIM DB: Get next pseudonym.

Parameters:

priv Private data pointer from eap_sim_db_init()

aka Using EAP-AKA instead of EAP-SIM

Returns:

Next pseudonym (allocated string) or NULL on failure

This function is used to generate a pseudonym for EAP-SIM. The returned pseudonym is not added to database at this point; it will need to be added with eap_sim_db_add_pseudonym() once the authentication has been completed successfully. Caller is responsible for freeing the returned buffer.

15.259.2.7 char* eap_sim_db_get_next_reauth_id (void * *priv*, int *aka*)

EAP-SIM DB: Get next reauth_id.

Parameters:

priv Private data pointer from eap_sim_db_init()

aka Using EAP-AKA instead of EAP-SIM

Returns:

Next reauth_id (allocated string) or NULL on failure

This function is used to generate a fast re-authentication identity for EAP-SIM. The returned reauth_id is not added to database at this point; it will need to be added with eap_sim_db_add_reauth() once the authentication has been completed successfully. Caller is responsible for freeing the returned buffer.

15.259.2.8 const u8* eap_sim_db_get_permanent (void * *priv*, const u8 * *identity*, size_t *identity_len*, size_t * *len*)

EAP-SIM DB: Get permanent identity.

Parameters:

priv Private data pointer from eap_sim_db_init()

identity Identity of the user (may be permanent identity or pseudonym)

identity_len Length of identity

len Buffer for length of the returned permanent identity

Returns:

Pointer to the permanent identity, or NULL if not found

15.259.2.9 struct eap_sim_reauth* eap_sim_db_get_reauth_entry (void * *priv*, const u8 * *identity*, size_t *identity_len*) [read]

EAP-SIM DB: Get re-authentication entry.

Parameters:

priv Private data pointer from eap_sim_db_init()

identity Identity of the user (may be permanent identity, pseudonym, or reauth_id)

identity_len Length of identity

Returns:

Pointer to the re-auth entry, or NULL if not found

15.259.2.10 int eap_sim_db_identity_known (void * *priv*, const u8 * *identity*, size_t *identity_len*)

Verify whether the given identity is known.

Parameters:

priv Private data pointer from eap_sim_db_init()

identity User name identity

identity_len Length of identity in bytes

Returns:

0 if the user is found or -1 on failure

In most cases, the user name is ['0','1'] | IMSI, i.e., 1 followed by the IMSI in ASCII format, ['2','3'] | pseudonym, or ['4','5'] | reauth_id.

15.259.2.11 void* eap_sim_db_init (const char * *config*, void(*)(void *ctx, void *session_ctx) *get_complete_cb*, void * *ctx*)

Initialize EAP-SIM DB / authentication gateway interface.

Parameters:

config Configuration data (e.g., file name)

get_complete_cb Callback function for reporting availability of triplets

ctx Context pointer for get_complete_cb

Returns:

Pointer to a private data structure or NULL on failure

15.259.2.12 void eap_sim_db_remove_reauth (void * *priv*, struct eap_sim_reauth * *reauth*)

EAP-SIM DB: Remove re-authentication entry.

Parameters:

priv Private data pointer from eap_sim_db_init()

reauth Pointer to re-authentication entry from eap_sim_db_get_reauth_entry()

15.259.2.13 int eap_sim_db_resynchronize (void * *priv*, const u8 * *identity*, size_t *identity_len*, const u8 * *auts*, const u8 * *_rand*)

Resynchronize AKA AUTN.

Parameters:

priv Private data pointer from eap_sim_db_init()

identity User name identity

identity_len Length of identity in bytes

auts AUTS value from the peer

_rand RAND value used in the rejected message

Returns:

0 on success, -1 on failure

This function is called when the peer reports synchronization failure in the AUTN value by sending AUTS. The AUTS and RAND values should be sent to HLR/AuC to allow it to resynchronize with the peer. After this, eap_sim_db_get_aka_auth() will be called again to to fetch updated RAND/AUTN values for the next challenge.

15.260 src/eap_server/eap_sim_db.h File Reference

hostapd / EAP-SIM database/authenticator gateway

15.260.1 Detailed Description

hostapd / EAP-SIM database/authenticator gateway

Copyright

15.261 src/eap_server/tncs.c File Reference

EAP-TNC - TNCS (IF-IMV, IF-TNCCS, and IF-TNCCS-SOH). `#include "includes.h"`

```
#include <dlfcn.h>
#include "common.h"
#include "base64.h"
#include "tncs.h"
#include "eap_common/eap_tlv_common.h"
#include "eap_common/eap_defs.h"
```

Data Structures

- struct tnc_if_imv
- struct tncs_data
- struct tncs_data::conn_imv
- struct tncs_global

Defines

- #define **TNC_CONFIG_FILE** "/etc/tnc_config"
- #define **IF_TNCCS_START**
- #define **IF_TNCCS_END** "\n</TNCCS-Batch>"
- #define **TNC_RESULT_SUCCESS** 0
- #define **TNC_RESULT_NOT_INITIALIZED** 1
- #define **TNC_RESULT_ALREADY_INITIALIZED** 2
- #define **TNC_RESULT_NO_COMMON_VERSION** 3
- #define **TNC_RESULT_CANT_RETRY** 4
- #define **TNC_RESULT_WONT_RETRY** 5
- #define **TNC_RESULT_INVALID_PARAMETER** 6
- #define **TNC_RESULT_CANT_RESPOND** 7
- #define **TNC_RESULT_ILLEGAL_OPERATION** 8
- #define **TNC_RESULT_OTHER** 9
- #define **TNC_RESULT_FATAL** 10
- #define **TNC_CONNECTION_STATE_CREATE** 0
- #define **TNC_CONNECTION_STATE_HANDSHAKE** 1
- #define **TNC_CONNECTION_STATE_ACCESS_ALLOWED** 2
- #define **TNC_CONNECTION_STATE_ACCESS_ISOLATED** 3
- #define **TNC_CONNECTION_STATE_ACCESS_NONE** 4
- #define **TNC_CONNECTION_STATE_DELETE** 5
- #define **TNC_IFIMV_VERSION_1** 1
- #define **TNC_VENDORID_ANY** ((TNC_VendorID) 0xffffff)
- #define **TNC_SUBTYPE_ANY** ((TNC_Subtype) 0xff)
- #define **TNC_TNCCS_RECOMMENDATION** 0x00000001
- #define **TNC_TNCCS_ERROR** 0x00000002
- #define **TNC_TNCCS_PREFERREDLANGUAGE** 0x00000003
- #define **TNC_TNCCS_REASONSTRINGS** 0x00000004
- #define **TNC_MAX_IMV_ID** 10

Typedefs

- typedef unsigned long **TNC_UInt32**
- typedef unsigned char * **TNC_BufferReference**
- typedef TNC_UInt32 **TNC_IMVID**
- typedef TNC_UInt32 **TNC_ConnectionID**
- typedef TNC_UInt32 **TNC_ConnectionState**
- typedef TNC_UInt32 **TNC_RetryReason**
- typedef TNC_UInt32 **TNC_IMV_Action_Recommendation**
- typedef TNC_UInt32 **TNC_IMV_Evaluation_Result**
- typedef TNC_UInt32 **TNC_MessageType**
- typedef TNC_MessageType * **TNC_MessageTypeList**
- typedef TNC_UInt32 **TNC_VendorID**
- typedef TNC_UInt32 **TNC_Subtype**
- typedef TNC_UInt32 **TNC_Version**
- typedef TNC_UInt32 **TNC_Result**
- typedef TNC_UInt32 **TNC_AttributeID**
- typedef TNC_Result(* **TNC_TNCS_BindFunctionPointer**)(TNC_IMVID imvID, char *functionName, void **pOutfunctionPointer)

Enumerations

- enum **IMV_Action_Recommendation** { **TNC_IMV_ACTION_RECOMMENDATION_ALLOW**, **TNC_IMV_ACTION_RECOMMENDATION_NO_ACCESS**, **TNC_IMV_ACTION_RECOMMENDATION_ISOLATE**, **TNC_IMV_ACTION_RECOMMENDATION_NO_RECOMMENDATION** }
- enum **IMV_Evaluation_Result** {

  TNC_IMV_EVALUATION_RESULT_COMPLIANT, **TNC_IMV_EVALUATION_RESULT_NONCOMPLIANT_MINOR**, **TNC_IMV_EVALUATION_RESULT_NONCOMPLIANT_MAJOR**, **TNC_IMV_EVALUATION_RESULT_ERROR**,

  TNC_IMV_EVALUATION_RESULT_DONT_KNOW }

Functions

- TNC_Result **TNC_TNCS_ReportMessageTypes** (TNC_IMVID imvID, TNC_MessageTypeList supportedTypes, TNC_UInt32 typeCount)
- TNC_Result **TNC_TNCS_SendMessage** (TNC_IMVID imvID, TNC_ConnectionID connectionID, TNC_BufferReference message, TNC_UInt32 messageLength, TNC_MessageType messageType)
- TNC_Result **TNC_TNCS_RequestHandshakeRetry** (TNC_IMVID imvID, TNC_ConnectionID connectionID, TNC_RetryReason reason)
- TNC_Result **TNC_TNCS_ProvideRecommendation** (TNC_IMVID imvID, TNC_ConnectionID connectionID, TNC_IMV_Action_Recommendation recommendation, TNC_IMV_Evaluation_Result evaluation)
- TNC_Result **TNC_TNCS_GetAttribute** (TNC_IMVID imvID, TNC_ConnectionID connectionID, TNC_AttributeID attribureID, TNC_UInt32 bufferLength, TNC_BufferReference buffer, TNC_UInt32 *pOutValueLength)
- TNC_Result **TNC_TNCS_SetAttribute** (TNC_IMVID imvID, TNC_ConnectionID connectionID, TNC_AttributeID attribureID, TNC_UInt32 bufferLength, TNC_BufferReference buffer)

- TNC_Result **TNC_TNCS_BindFunction** (TNC_IMVID imvID, char *functionName, void **pOutFunctionPointer)
- void **tncs_init_connection** (struct tncs_data *tncs)
- size_t **tncs_total_send_len** (struct tncs_data *tncs)
- u8 * **tncs_copy_send_buf** (struct tncs_data *tncs, u8 *pos)
- char * **tncs_if_tnccs_start** (struct tncs_data *tncs)
- char * **tncs_if_tnccs_end** (void)
- enum tncs_process_res **tncs_process_if_tnccs** (struct tncs_data *tncs, const u8 *msg, size_t len)
- struct tncs_data * **tncs_init** (void)
- void **tncs_deinit** (struct tncs_data *tncs)
- int **tncs_global_init** (void)
- void **tncs_global_deinit** (void)
- struct wpabuf * **tncs_build_soh_request** (void)
- struct wpabuf * **tncs_process_soh** (const u8 *soh_tlv, size_t soh_tlv_len, int *failure)

15.261.1 Detailed Description

EAP-TNC - TNCS (IF-IMV, IF-TNCCS, and IF-TNCCS-SOH).

Copyright

Copyright (c) 2007-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.261.2 Define Documentation

15.261.2.1 #define IF_TNCCS_START

Value:

```
"<?xml version=\"1.0\"?>\n" \
"<TNCCS-Batch BatchId=\"%d\" Recipient=\"TNCS\" " \
"xmlns=\"http://www.trustedcomputinggroup.org/IWG/TNC/1_0/IF_TNCCS#\" " \
"xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" " \
"xsi:schemaLocation=\"http://www.trustedcomputinggroup.org/IWG/TNC/1_0/" \
"IF_TNCCS# https://www.trustedcomputinggroup.org/XML/SCHEMA/TNCCS_1.0.xsd\">\n"
```

15.262 src/eap_server/tncs.h File Reference

EAP-TNC - TNCS (IF-IMV, IF-TNCCS, and IF-TNCCS-SOH).

Enumerations

- enum **tncs_process_res** {

  TNCCS_PROCESS_ERROR = -1, **TNCCS_PROCESS_OK_NO_RECOMMENDATION** = 0, **TNCCS_RECOMMENDATION_ERROR**, **TNCCS_RECOMMENDATION_ALLOW**,

  TNCCS_RECOMMENDATION_NONE, **TNCCS_RECOMMENDATION_ISOLATE**, **TNCCS_RECOMMENDATION_NO_ACCESS**, **TNCCS_RECOMMENDATION_NO_-RECOMMENDATION** }

Functions

- struct tncs_data * **tncs_init** (void)
- void **tncs_deinit** (struct tncs_data *tncs)
- void **tncs_init_connection** (struct tncs_data *tncs)
- size_t **tncs_total_send_len** (struct tncs_data *tncs)
- u8 * **tncs_copy_send_buf** (struct tncs_data *tncs, u8 *pos)
- char * **tncs_if_tnccs_start** (struct tncs_data *tncs)
- char * **tncs_if_tnccs_end** (void)
- enum tncs_process_res **tncs_process_if_tnccs** (struct tncs_data *tncs, const u8 *msg, size_t len)
- int **tncs_global_init** (void)
- void **tncs_global_deinit** (void)
- struct wpabuf * **tncs_build_soh_request** (void)
- struct wpabuf * **tncs_process_soh** (const u8 *soh_tlv, size_t soh_tlv_len, int *failure)

15.262.1 Detailed Description

EAP-TNC - TNCS (IF-IMV, IF-TNCCS, and IF-TNCCS-SOH).

Copyright

15.263 src/eapol_supp/eapol_supp_sm.c File Reference

EAPOL supplicant state machines. `#include "includes.h"`

```
#include "common.h"
#include "eapol_supp_sm.h"
#include "eap_peer/eap.h"
#include "eloop.h"
#include "eapol_common.h"
#include "md5.h"
#include "crypto.h"
#include "state_machine.h"
#include "wpabuf.h"
```

Data Structures

- struct eapol_sm

  Internal data for EAPOL state machines.

- struct ieee802_1x_eapol_key
- struct eap_key_data

Defines

- #define **STATE_MACHINE_DATA** struct eapol_sm
- #define **STATE_MACHINE_DEBUG_PREFIX** "EAPOL"
- #define **IEEE8021X_REPLAY_COUNTER_LEN** 8
- #define **IEEE8021X_KEY_SIGN_LEN** 16
- #define **IEEE8021X_KEY_IV_LEN** 16
- #define **IEEE8021X_KEY_INDEX_FLAG** 0x80
- #define **IEEE8021X_KEY_INDEX_MASK** 0x03
- #define **IEEE8021X_ENCR_KEY_LEN** 32
- #define **IEEE8021X_SIGN_KEY_LEN** 32

Functions

- **SM_STATE** (SUPP_PAE, LOGOFF)
- **SM_STATE** (SUPP_PAE, DISCONNECTED)
- **SM_STATE** (SUPP_PAE, CONNECTING)
- **SM_STATE** (SUPP_PAE, AUTHENTICATING)
- **SM_STATE** (SUPP_PAE, HELD)
- **SM_STATE** (SUPP_PAE, AUTHENTICATED)
- **SM_STATE** (SUPP_PAE, RESTART)
- **SM_STATE** (SUPP_PAE, S_FORCE_AUTH)
- **SM_STATE** (SUPP_PAE, S_FORCE_UNAUTH)
- **SM_STEP** (SUPP_PAE)

- **SM_STATE** (KEY_RX, NO_KEY_RECEIVE)
- **SM_STATE** (KEY_RX, KEY_RECEIVE)
- **SM_STEP** (KEY_RX)
- **SM_STATE** (SUPP_BE, REQUEST)
- **SM_STATE** (SUPP_BE, RESPONSE)
- **SM_STATE** (SUPP_BE, SUCCESS)
- **SM_STATE** (SUPP_BE, FAIL)
- **SM_STATE** (SUPP_BE, TIMEOUT)
- **SM_STATE** (SUPP_BE, IDLE)
- **SM_STATE** (SUPP_BE, INITIALIZE)
- **SM_STATE** (SUPP_BE, RECEIVE)
- **SM_STEP** (SUPP_BE)
- void eapol_sm_step (struct eapol_sm *sm)

  EAPOL state machine step function.

- void eapol_sm_configure (struct eapol_sm *sm, int heldPeriod, int authPeriod, int startPeriod, int maxStart)

  Set EAPOL variables.

- int eapol_sm_get_status (struct eapol_sm *sm, char *buf, size_t buflen, int verbose)

  Get EAPOL state machine status.

- int eapol_sm_get_mib (struct eapol_sm *sm, char *buf, size_t buflen)

  Get EAPOL state machine MIBs.

- int eapol_sm_rx_eapol (struct eapol_sm *sm, const u8 *src, const u8 *buf, size_t len)

  Process received EAPOL frames.

- void eapol_sm_notify_tx_eapol_key (struct eapol_sm *sm)

  Notification about transmitted EAPOL packet.

- void eapol_sm_notify_portEnabled (struct eapol_sm *sm, Boolean enabled)

  Notification about portEnabled change.

- void eapol_sm_notify_portValid (struct eapol_sm *sm, Boolean valid)

  Notification about portValid change.

- void eapol_sm_notify_eap_success (struct eapol_sm *sm, Boolean success)

  Notification of external EAP success trigger.

- void eapol_sm_notify_eap_fail (struct eapol_sm *sm, Boolean fail)

  Notification of external EAP failure trigger.

- void eapol_sm_notify_config (struct eapol_sm *sm, struct eap_peer_config *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, int in_eapol_sm)
  Notification of lower layer success.
- void eapol_sm_invalidate_cached_session (struct eapol_sm *sm)
  Mark cached EAP session data invalid.
- struct 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

- struct ieee802_1x_eapol_key **STRUCT_PACKED**

15.263.1 Detailed Description

EAPOL supplicant state machines.

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.263.2 Function Documentation

15.263.2.1 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).

15.263.2.2 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.

15.263.2.3 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.

15.263.2.4 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.

15.263.2.5 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.

15.263.2.6 struct eapol_sm* eapol_sm_init (struct eapol_ctx * *ctx*) [read]

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.

15.263.2.7 void eapol_sm_invalidate_cached_session (struct eapol_sm * *sm*)

Mark cached EAP session data invalid.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

15.263.2.8 void eapol_sm_notify_cached (struct eapol_sm * *sm*)

Notification of successful PMKSA caching.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that PMKSA caching was successful. This is used to move EAPOL and EAP state machines into authenticated/successful state.

15.263.2.9 void eapol_sm_notify_config (struct eapol_sm * *sm*, struct eap_peer_config * *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 EAP configuration

conf Pointer to EAPOL configuration data

Notify EAPOL state machine that configuration has changed. config will be stored as a backpointer to network configuration. This can be NULL to clear the stored pointed. conf will be copied to local EAPOL/EAP configuration data. If conf is NULL, this part of the configuration change will be skipped.

15.263.2.10 void eapol_sm_notify_ctrl_attached (struct eapol_sm * *sm*)

Notification of attached monitor.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that a monitor was attached to the control interface to trigger re-sending of pending requests for user input.

15.263.2.11 void eapol_sm_notify_ctrl_response (struct eapol_sm * *sm*)

Notification of received user input.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that a control response, i.e., user input, was received in order to trigger retrying of a pending EAP request.

15.263.2.12 void eapol_sm_notify_eap_fail (struct eapol_sm * *sm*, Boolean *fail*)

Notification of external EAP failure trigger.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

fail TRUE = set failure, FALSE = clear failure

Notify EAPOL state machine that external event has forced EAP state to failure (fail = TRUE). This can be cleared by setting fail = FALSE.

15.263.2.13 void eapol_sm_notify_eap_success (struct eapol_sm * *sm*, Boolean *success*)

Notification of external EAP success trigger.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

success TRUE = set success, FALSE = clear success

Notify the EAPOL state 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.

15.263.2.14 void eapol_sm_notify_logoff (struct eapol_sm * *sm*, Boolean *logoff*)

Notification of logon/logoff commands.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

logoff Whether command was logoff

Notify EAPOL state machines that user requested logon/logoff.

15.263.2.15 void eapol_sm_notify_lower_layer_success (struct eapol_sm * *sm*, int *in_eapol_sm*)

Notification of lower layer success.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

in_eapol_sm Whether the caller is already running inside EAPOL state machine loop (eapol_sm_step())

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.

15.263.2.16 void eapol_sm_notify_pmkid_attempt (struct eapol_sm * *sm*, int *attempt*)

Notification of PMKSA caching.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()
attempt Whether PMKSA caching is tried

Notify EAPOL state machines whether PMKSA caching is used.

15.263.2.17 void eapol_sm_notify_portControl (struct eapol_sm * *sm*, PortControl *portControl*)

Notification of portControl changes.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()
portControl New value for portControl variable

Notify EAPOL state machines that portControl variable has changed.

15.263.2.18 void eapol_sm_notify_portEnabled (struct eapol_sm * *sm*, Boolean *enabled*)

Notification about portEnabled change.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()
enabled New portEnabled value

Notify EAPOL state machine about new portEnabled value.

15.263.2.19 void eapol_sm_notify_portValid (struct eapol_sm * *sm*, Boolean *valid*)

Notification about portValid change.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()
valid New portValid value

Notify EAPOL state machine about new portValid value.

15.263.2.20 void eapol_sm_notify_tx_eapol_key (struct eapol_sm * *sm*)

Notification about transmitted EAPOL packet.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machine about transmitted EAPOL packet from an external component, e.g., WPA. This will update the statistics.

15.263.2.21 void eapol_sm_register_scard_ctx (struct eapol_sm * *sm*, void * *ctx*)

Notification of smart card context.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

ctx Context data for smart card operations

Notify EAPOL state machines of context data for smart card operations. This context data will be used as a parameter for scard_*() functions.

15.263.2.22 void eapol_sm_request_reauth (struct eapol_sm * *sm*)

Request reauthentication.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

This function can be used to request EAPOL reauthentication, e.g., when the current PMKSA entry is nearing expiration.

15.263.2.23 int eapol_sm_rx_eapol (struct eapol_sm * *sm*, const u8 * *src*, const u8 * *buf*, size_t *len*)

Process received EAPOL frames.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

src Source MAC address of the EAPOL packet

buf Pointer to the beginning of the EAPOL data (EAPOL header)

len Length of the EAPOL frame

Returns:

1 = EAPOL frame processed, 0 = not for EAPOL state machine, -1 failure

15.263.2.24 void eapol_sm_step (struct eapol_sm * *sm*)

EAPOL state machine step function.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

This function is called to notify the state machine about changed external variables. It will step through the EAPOL state machines in loop to process all triggered state changes.

15.264 src/eapol_supp/eapol_supp_sm.h File Reference

EAPOL supplicant state machines. `#include "defs.h"`

Data Structures

- struct eapol_config

  Per network configuration for EAPOL state machines.

- struct eapol_ctx

  Global (for all networks) EAPOL state machine context.

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

- struct 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 eap_peer_config *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, int in_eapol_sm)
  Notification of lower layer success.
- void eapol_sm_invalidate_cached_session (struct eapol_sm *sm)
  Mark cached EAP session data invalid.

15.264.1 Detailed Description

EAPOL supplicant state machines.

Copyright

Copyright (c) 2004-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.264.2 Function Documentation

15.264.2.1 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).

15.264.2.2 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.

15.264.2.3 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.

15.264.2.4 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.

15.264.2.5 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.

15.264.2.6 struct eapol_sm* eapol_sm_init (struct eapol_ctx * *ctx*) `[read]`

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.

15.264.2.7 void eapol_sm_invalidate_cached_session (struct eapol_sm * *sm*)

Mark cached EAP session data invalid.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

15.264.2.8 void eapol_sm_notify_cached (struct eapol_sm * *sm*)

Notification of successful PMKSA caching.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that PMKSA caching was successful. This is used to move EAPOL and EAP state machines into authenticated/successful state.

15.264.2.9 void eapol_sm_notify_config (struct eapol_sm * *sm*, struct eap_peer_config * *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 EAP configuration

conf Pointer to EAPOL configuration data

Notify EAPOL state machine that configuration has changed. config will be stored as a backpointer to network configuration. This can be NULL to clear the stored pointed. conf will be copied to local EAPOL/EAP configuration data. If conf is NULL, this part of the configuration change will be skipped.

15.264.2.10 void eapol_sm_notify_ctrl_attached (struct eapol_sm * *sm*)

Notification of attached monitor.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that a monitor was attached to the control interface to trigger re-sending of pending requests for user input.

15.264.2.11 void eapol_sm_notify_ctrl_response (struct eapol_sm * *sm*)

Notification of received user input.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machines that a control response, i.e., user input, was received in order to trigger retrying of a pending EAP request.

15.264.2.12 void eapol_sm_notify_eap_fail (struct eapol_sm * *sm*, Boolean *fail*)

Notification of external EAP failure trigger.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

fail TRUE = set failure, FALSE = clear failure

Notify EAPOL state machine that external event has forced EAP state to failure (fail = TRUE). This can be cleared by setting fail = FALSE.

15.264.2.13 void eapol_sm_notify_eap_success (struct eapol_sm * *sm*, Boolean *success*)

Notification of external EAP success trigger.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

success TRUE = set success, FALSE = clear success

Notify the EAPOL state 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.

15.264.2.14 void eapol_sm_notify_logoff (struct eapol_sm * *sm*, Boolean *logoff*)

Notification of logon/logoff commands.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

logoff Whether command was logoff

Notify EAPOL state machines that user requested logon/logoff.

15.264.2.15 void eapol_sm_notify_lower_layer_success (struct eapol_sm * *sm*, int *in_eapol_sm*)

Notification of lower layer success.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

in_eapol_sm Whether the caller is already running inside EAPOL state machine loop (eapol_sm_step())

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.

15.264.2.16 void eapol_sm_notify_pmkid_attempt (struct eapol_sm * *sm*, int *attempt*)

Notification of PMKSA caching.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

attempt Whether PMKSA caching is tried

Notify EAPOL state machines whether PMKSA caching is used.

15.264.2.17 void eapol_sm_notify_portControl (struct eapol_sm * *sm*, PortControl *portControl*)

Notification of portControl changes.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

portControl New value for portControl variable

Notify EAPOL state machines that portControl variable has changed.

15.264.2.18 void eapol_sm_notify_portEnabled (struct eapol_sm * *sm*, Boolean *enabled*)

Notification about portEnabled change.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

enabled New portEnabled value

Notify EAPOL state machine about new portEnabled value.

15.264.2.19 void eapol_sm_notify_portValid (struct eapol_sm * *sm*, Boolean *valid*)

Notification about portValid change.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

valid New portValid value

Notify EAPOL state machine about new portValid value.

15.264.2.20 void eapol_sm_notify_tx_eapol_key (struct eapol_sm * *sm*)

Notification about transmitted EAPOL packet.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

Notify EAPOL state machine about transmitted EAPOL packet from an external component, e.g., WPA. This will update the statistics.

15.264.2.21 void eapol_sm_register_scard_ctx (struct eapol_sm * *sm*, void * *ctx*)

Notification of smart card context.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

ctx Context data for smart card operations

Notify EAPOL state machines of context data for smart card operations. This context data will be used as a parameter for scard_*() functions.

15.264.2.22 void eapol_sm_request_reauth (struct eapol_sm * *sm*)

Request reauthentication.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

This function can be used to request EAPOL reauthentication, e.g., when the current PMKSA entry is nearing expiration.

15.264.2.23 int eapol_sm_rx_eapol (struct eapol_sm * *sm*, const u8 * *src*, const u8 * *buf*, size_t *len*)

Process received EAPOL frames.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

src Source MAC address of the EAPOL packet

buf Pointer to the beginning of the EAPOL data (EAPOL header)

len Length of the EAPOL frame

Returns:

1 = EAPOL frame processed, 0 = not for EAPOL state machine, -1 failure

15.264.2.24 void eapol_sm_step (struct eapol_sm * *sm*)

EAPOL state machine step function.

Parameters:

sm Pointer to EAPOL state machine allocated with eapol_sm_init()

This function is called to notify the state machine about changed external variables. It will step through the EAPOL state machines in loop to process all triggered state changes.

15.265 src/hlr_auc_gw/hlr_auc_gw.c File Reference

HLR/AuC testing gateway for hostapd EAP-SIM/AKA database/authenticator. `#include "includes.h"`

`#include <sys/un.h>`

`#include "common.h"`

`#include "milenage.h"`

Data Structures

- struct gsm_triplet
- struct milenage_parameters

Defines

- #define **EAP_SIM_MAX_CHAL** 3
- #define **EAP_AKA_RAND_LEN** 16
- #define **EAP_AKA_AUTN_LEN** 16
- #define **EAP_AKA_AUTS_LEN** 14
- #define **EAP_AKA_RES_MAX_LEN** 16
- #define **EAP_AKA_IK_LEN** 16
- #define **EAP_AKA_CK_LEN** 16

Functions

- int **main** (int argc, char ∗argv[ ])

15.265.1 Detailed Description

HLR/AuC testing gateway for hostapd EAP-SIM/AKA database/authenticator.

Copyright

This is an example implementation of the EAP-SIM/AKA database/authentication gateway interface to HLR/AuC. It is expected to be replaced with an implementation of SS7 gateway to GSM/UMTS authentication center (HLR/AuC) or a local implementation of SIM triplet and AKA authentication data generator.

hostapd will send SIM/AKA authentication queries over a UNIX domain socket to and external program, e.g., this hlr_auc_gw. This interface uses simple text-based format:

EAP-SIM / GSM triplet query/response: SIM-REQ-AUTH <IMSI> <max_chal> SIM-RESP-AUTH <IMSI> Kc1:SRES1:RAND1 Kc2:SRES2:RAND2 [Kc3:SRES3:RAND3] SIM-RESP-AUTH <IMSI> FAILURE

EAP-AKA / UMTS query/response: AKA-REQ-AUTH <IMSI> AKA-RESP-AUTH <IMSI> <RAND> <AUTN> <IK> <CK> <RES> AKA-RESP-AUTH <IMSI> FAILURE

EAP-AKA / UMTS AUTS (re-synchronization): AKA-AUTS <IMSI> <AUTS> <RAND>

IMSI and max_chal are sent as an ASCII string, Kc/SRES/RAND/AUTN/IK/CK/RES/AUTS as hex strings.

The example implementation here reads GSM authentication triplets from a text file in IMSI:Kc:SRES:RAND format, IMSI in ASCII, other fields as hex strings. This is used to simulate an HLR/AuC. As such, it is not very useful for real life authentication, but it is useful both as an example implementation and for EAP-SIM testing.

15.266 src/hlr_auc_gw/milenage.c File Reference

3GPP AKA - Milenage algorithm (3GPP TS 35.205, .206, .207, .208) `#include "includes.h"`

`#include "common.h"`

`#include "milenage.h"`

`#include "aes_wrap.h"`

Functions

- void milenage_generate (const u8 *opc, const u8 *amf, const u8 *k, const u8 *sqn, const u8 *_rand, u8 *autn, u8 *ik, u8 *ck, u8 *res, size_t *res_len)
  Generate AKA AUTN,IK,CK,RES.
- int milenage_auts (const u8 *opc, const u8 *k, const u8 *_rand, const u8 *auts, u8 *sqn)
  Milenage AUTS validation.
- int gsm_milenage (const u8 *opc, const u8 *k, const u8 *_rand, u8 *sres, u8 *kc)
  Generate GSM-Milenage (3GPP TS 55.205) authentication triplet.
- int milenage_check (const u8 *opc, const u8 *k, const u8 *sqn, const u8 *_rand, const u8 *autn, u8 *ik, u8 *ck, u8 *res, size_t *res_len, u8 *auts)
  Generate AKA AUTN,IK,CK,RES.

15.266.1 Detailed Description

3GPP AKA - Milenage algorithm (3GPP TS 35.205, .206, .207, .208)

Copyright

Copyright (c) 2006-2007 <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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 an example authentication algorithm defined for 3GPP AKA. This can be used to implement a simple HLR/AuC into hlr_auc_gw to allow EAP-AKA to be tested properly with real USIM cards.

This implementations assumes that the r1..r5 and c1..c5 constants defined in TS 35.206 are used, i.e., r1=64, r2=0, r3=32, r4=64, r5=96, c1=00..00, c2=00..01, c3=00..02, c4=00..04, c5=00..08. The block cipher is assumed to be AES (Rijndael).

15.266.2 Function Documentation

15.266.2.1 int gsm_milenage (const u8 * *opc*, const u8 * *k*, const u8 * *_rand*, u8 * *sres*, u8 * *kc*)

Generate GSM-Milenage (3GPP TS 55.205) authentication triplet.

Parameters:

opc OPc = 128-bit operator variant algorithm configuration field (encr.)

k K = 128-bit subscriber key

_rand RAND = 128-bit random challenge

sres Buffer for SRES = 32-bit SRES

kc Buffer for Kc = 64-bit Kc

Returns:

0 on success, -1 on failure

15.266.2.2 int milenage_auts (const u8 * *opc*, const u8 * *k*, const u8 * *_rand*, const u8 * *auts*, u8 * *sqn*)

Milenage AUTS validation.

Parameters:

opc OPc = 128-bit operator variant algorithm configuration field (encr.)

k K = 128-bit subscriber key

_rand RAND = 128-bit random challenge

auts AUTS = 112-bit authentication token from client

sqn Buffer for SQN = 48-bit sequence number

Returns:

0 = success (sqn filled), -1 on failure

15.266.2.3 int milenage_check (const u8 * *opc*, const u8 * *k*, const u8 * *sqn*, const u8 * *_rand*, const u8 * *autn*, u8 * *ik*, u8 * *ck*, u8 * *res*, size_t * *res_len*, u8 * *auts*)

Generate AKA AUTN,IK,CK,RES.

Parameters:

opc OPc = 128-bit operator variant algorithm configuration field (encr.)

k K = 128-bit subscriber key

sqn SQN = 48-bit sequence number

_rand RAND = 128-bit random challenge

autn AUTN = 128-bit authentication token

ik Buffer for IK = 128-bit integrity key (f4), or NULL

ck Buffer for CK = 128-bit confidentiality key (f3), or NULL

res Buffer for RES = 64-bit signed response (f2), or NULL

res_len Variable that will be set to RES length

auts 112-bit buffer for AUTS

Returns:

0 on success, -1 on failure, or -2 on synchronization failure

15.266.2.4 void milenage_generate (const u8 * *opc*, const u8 * *amf*, const u8 * *k*, const u8 * *sqn*, const u8 * *_rand*, u8 * *autn*, u8 * *ik*, u8 * *ck*, u8 * *res*, size_t * *res_len*)

Generate AKA AUTN,IK,CK,RES.

Parameters:

opc OPc = 128-bit operator variant algorithm configuration field (encr.)

amf AMF = 16-bit authentication management field

k K = 128-bit subscriber key

sqn SQN = 48-bit sequence number

_rand RAND = 128-bit random challenge

autn Buffer for AUTN = 128-bit authentication token

ik Buffer for IK = 128-bit integrity key (f4), or NULL

ck Buffer for CK = 128-bit confidentiality key (f3), or NULL

res Buffer for RES = 64-bit signed response (f2), or NULL

res_len Max length for res; set to used length or 0 on failure

15.267 src/hlr_auc_gw/milenage.h File Reference

UMTS AKA - Milenage algorithm (3GPP TS 35.205, .206, .207, .208).

Functions

- void milenage_generate (const u8 *opc, const u8 *amf, const u8 *k, const u8 *sqn, const u8 *_rand, u8 *autn, u8 *ik, u8 *ck, u8 *res, size_t *res_len)
  Generate AKA AUTN,IK,CK,RES.
- int milenage_auts (const u8 *opc, const u8 *k, const u8 *_rand, const u8 *auts, u8 *sqn)
  Milenage AUTS validation.
- int gsm_milenage (const u8 *opc, const u8 *k, const u8 *_rand, u8 *sres, u8 *kc)
  Generate GSM-Milenage (3GPP TS 55.205) authentication triplet.
- int milenage_check (const u8 *opc, const u8 *k, const u8 *sqn, const u8 *_rand, const u8 *autn, u8 *ik, u8 *ck, u8 *res, size_t *res_len, u8 *auts)
  Generate AKA AUTN,IK,CK,RES.

15.267.1 Detailed Description

UMTS AKA - Milenage algorithm (3GPP TS 35.205, .206, .207, .208).

Copyright

Copyright (c) 2006-2007 <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.267.2 Function Documentation

15.267.2.1 int gsm_milenage (const u8 * *opc*, const u8 * *k*, const u8 * *_rand*, u8 * *sres*, u8 * *kc*)

Generate GSM-Milenage (3GPP TS 55.205) authentication triplet.

Parameters:

opc OPc = 128-bit operator variant algorithm configuration field (encr.)
k K = 128-bit subscriber key
_rand RAND = 128-bit random challenge
sres Buffer for SRES = 32-bit SRES
kc Buffer for Kc = 64-bit Kc

Returns:

0 on success, -1 on failure

15.267.2.2 int milenage_auts (const u8 * *opc*, const u8 * *k*, const u8 * *_rand*, const u8 * *auts*, u8 * *sqn*)

Milenage AUTS validation.

Parameters:

opc OPc = 128-bit operator variant algorithm configuration field (encr.)

k K = 128-bit subscriber key

_rand RAND = 128-bit random challenge

auts AUTS = 112-bit authentication token from client

sqn Buffer for SQN = 48-bit sequence number

Returns:

0 = success (sqn filled), -1 on failure

15.267.2.3 int milenage_check (const u8 * *opc*, const u8 * *k*, const u8 * *sqn*, const u8 * *_rand*, const u8 * *autn*, u8 * *ik*, u8 * *ck*, u8 * *res*, size_t * *res_len*, u8 * *auts*)

Generate AKA AUTN,IK,CK,RES.

Parameters:

opc OPc = 128-bit operator variant algorithm configuration field (encr.)

k K = 128-bit subscriber key

sqn SQN = 48-bit sequence number

_rand RAND = 128-bit random challenge

autn AUTN = 128-bit authentication token

ik Buffer for IK = 128-bit integrity key (f4), or NULL

ck Buffer for CK = 128-bit confidentiality key (f3), or NULL

res Buffer for RES = 64-bit signed response (f2), or NULL

res_len Variable that will be set to RES length

auts 112-bit buffer for AUTS

Returns:

0 on success, -1 on failure, or -2 on synchronization failure

15.267.2.4 void milenage_generate (const u8 * *opc*, const u8 * *amf*, const u8 * *k*, const u8 * *sqn*, const u8 * *_rand*, u8 * *autn*, u8 * *ik*, u8 * *ck*, u8 * *res*, size_t * *res_len*)

Generate AKA AUTN,IK,CK,RES.

Parameters:

opc OPc = 128-bit operator variant algorithm configuration field (encr.)

amf AMF = 16-bit authentication management field

k K = 128-bit subscriber key

sqn SQN = 48-bit sequence number

_rand RAND = 128-bit random challenge

autn Buffer for AUTN = 128-bit authentication token

ik Buffer for IK = 128-bit integrity key (f4), or NULL

ck Buffer for CK = 128-bit confidentiality key (f3), or NULL

res Buffer for RES = 64-bit signed response (f2), or NULL

res_len Max length for res; set to used length or 0 on failure

15.268 src/l2_packet/l2_packet.h File Reference

WPA Supplicant - Layer2 packet interface definition.

Data Structures

- struct l2_ethhdr

Functions

- struct 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

- struct l2_ethhdr **STRUCT_PACKED**

15.268.1 Detailed Description

WPA Supplicant - Layer2 packet interface definition.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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.

15.268.2 Function Documentation

15.268.2.1 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()

15.268.2.2 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.

15.268.2.3 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

15.268.2.4 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*) [read]

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.

15.268.2.5 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.

15.268.2.6 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

15.269 src/l2_packet/l2_packet_freebsd.c File Reference

WPA Supplicant - Layer2 packet handling with FreeBSD. `#include "includes.h"`

```
#include <pcap.h>
#include <sys/ioctl.h>
#include <sys/sysctl.h>
#include <net/if.h>
#include <net/if_dl.h>
#include <net/route.h>
#include <netinet/in.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

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.

- struct 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.

15.269.1 Detailed Description

WPA Supplicant - Layer2 packet handling with FreeBSD.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi> Copyright (c) 2005, Sam Leffler <sam@errno.com>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.269.2 Function Documentation

15.269.2.1 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()

15.269.2.2 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.

15.269.2.3 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

15.269.2.4 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*) [read]

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.

15.269.2.5 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.

15.269.2.6 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

15.270 src/l2_packet/l2_packet_linux.c File Reference

WPA Supplicant - Layer2 packet handling with Linux packet sockets. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include <netpacket/packet.h>
#include <net/if.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

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.

- struct 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.

15.270.1 Detailed Description

WPA Supplicant - Layer2 packet handling with Linux packet sockets.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.270.2 Function Documentation

15.270.2.1 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()

15.270.2.2 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.

15.270.2.3 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

15.270.2.4 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*) [read]

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.

15.270.2.5 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.

15.270.2.6 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

15.271 src/l2_packet/l2_packet_ndis.c File Reference

WPA Supplicant - Layer2 packet handling with Microsoft NDISUIO. `#include "includes.h"`

```
#include <winsock2.h>
#include <ntddndis.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

Data Structures

- struct l2_packet_ndisuio_global
- struct l2_packet_data

Defines

- #define **FSCTL_NDISUIO_BASE** FILE_DEVICE_NETWORK
- #define **_NDISUIO_CTL_CODE**(_Function, _Method, _Access) CTL_CODE(FSCTL_NDISUIO_BASE, _Function, _Method, _Access)
- #define **IOCTL_NDISUIO_SET_ETHER_TYPE**

Functions

- HANDLE **driver_ndis_get_ndisuio_handle** (void)
- int l2_packet_get_own_addr (struct l2_packet_data *l2, u8 *addr)

  Get own layer 2 address.

- int l2_packet_send (struct l2_packet_data *l2, const u8 *dst_addr, u16 proto, const u8 *buf, size_t len)

  Send a packet.

- struct 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.

15.271.1 Detailed Description

WPA Supplicant - Layer2 packet handling with Microsoft NDISUIO.

Copyright

Copyright (c) 2003-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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 implementation requires Windows specific event loop implementation, i.e., eloop_win.c. In addition, the NDISUIO connection is shared with driver_ndis.c, so only that driver interface can be used and CONFIG_USE_NDISUIO must be defined.

WinXP version of the code uses overlapped I/O and a single threaded design with callback functions from I/O code. WinCE version uses a separate RX thread that blocks on ReadFile() whenever the media status is connected.

15.271.2 Define Documentation

15.271.2.1 #define IOCTL_NDISUIO_SET_ETHER_TYPE

Value:

```
_NDISUIO_CTL_CODE(0x202, METHOD_BUFFERED, \
                          FILE_READ_ACCESS | FILE_WRITE_ACCESS)
```

15.271.3 Function Documentation

15.271.3.1 void l2_packet_deinit (struct l2_packet_data * *l2*)

Deinitialize l2_packet interface.

Parameters:

l2 Pointer to internal l2_packet data from l2_packet_init()

15.271.3.2 int l2_packet_get_ip_addr (struct l2_packet_data * *l2*, char * *buf*, size_t *len*)

Get the current IP address from the interface.

Parameters:

l2 Pointer to internal l2_packet data from l2_packet_init()

buf Buffer for the IP address in text format

len Maximum buffer length

Returns:

0 on success, -1 on failure

This function can be used to get the current IP address from the interface bound to the l2_packet. This is mainly for status information and the IP address will be stored as an ASCII string. This function is not essential for wpa_supplicant operation, so full implementation is not required. l2_packet implementation will need to define the function, but it can return -1 if the IP address information is not available.

15.271.3.3 int l2_packet_get_own_addr (struct l2_packet_data * *l2*, u8 * *addr*)

Get own layer 2 address.

Parameters:

l2 Pointer to internal l2_packet data from l2_packet_init()

addr Buffer for the own address (6 bytes)

Returns:

0 on success, -1 on failure

15.271.3.4 struct l2_packet_data* l2_packet_init (const char * *ifname*, const u8 * *own_addr*, unsigned short *protocol*, void(*)(void *ctx, const u8 *src_addr, const u8 *buf, size_t len) *rx_callback*, void * *rx_callback_ctx*, int *l2_hdr*) [read]

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.

15.271.3.5 void l2_packet_notify_auth_start (struct l2_packet_data * *l2*)

Notify l2_packet about start of authentication.

Parameters:

l2 Pointer to internal l2_packet data from l2_packet_init()

This function is called when authentication is expected to start, e.g., when association has been completed, in order to prepare l2_packet implementation for EAPOL frames. This function is used mainly if the l2_packet code needs to do polling in which case it can increasing polling frequency. This can also be an empty function if the l2_packet implementation does not benefit from knowing about the starting authentication.

15.271.3.6 int l2_packet_send (struct l2_packet_data * *l2*, const u8 * *dst_addr*, u16 *proto*, const u8 * *buf*, size_t *len*)

Send a packet.

Parameters:

l2 Pointer to internal l2_packet data from l2_packet_init()

dst_addr Destination address for the packet (only used if l2_hdr == 0)

proto Protocol/ethertype for the packet in host byte order (only used if l2_hdr == 0)

buf Packet contents to be sent; including layer 2 header if l2_hdr was set to 1 in l2_packet_init() call. Otherwise, only the payload of the packet is included.

len Length of the buffer (including l2 header only if l2_hdr == 1)

Returns:

>=0 on success, <0 on failure

15.272 src/l2_packet/l2_packet_none.c File Reference

WPA Supplicant - Layer2 packet handling example with dummy functions. `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

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.

- struct 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.

15.272.1 Detailed Description

WPA Supplicant - Layer2 packet handling example with dummy functions.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file can be used as a starting point for layer2 packet implementation.

15.272.2 Function Documentation

15.272.2.1 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()

15.272.2.2 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.

15.272.2.3 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

15.272.2.4 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*) [read]

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.

15.272.2.5 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.

15.272.2.6 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

15.273 src/l2_packet/l2_packet_pcap.c File Reference

WPA Supplicant - Layer2 packet handling with libpcap/libdnet and WinPcap. `#include "includes.h"`

```
#include <sys/ioctl.h>
#include <pcap.h>
#include <dnet.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

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.

- struct 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.

15.273.1 Detailed Description

WPA Supplicant - Layer2 packet handling with libpcap/libdnet and WinPcap.

Copyright

Copyright (c) 2003-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.273.2 Function Documentation

15.273.2.1 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()

15.273.2.2 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.

15.273.2.3 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

15.273.2.4 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*) [read]

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.

15.273.2.5 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.

15.273.2.6 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

15.274 src/l2_packet/l2_packet_privsep.c File Reference

WPA Supplicant - Layer2 packet handling with privilege separation. `#include "includes.h"`

```
#include <sys/un.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
#include "privsep_commands.h"
```

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.

- struct 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.

15.274.1 Detailed Description

WPA Supplicant - Layer2 packet handling with privilege separation.

Copyright

Copyright (c) 2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.274.2 Function Documentation

15.274.2.1 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()

15.274.2.2 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.

15.274.2.3 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

15.274.2.4 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*) [read]

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.

15.274.2.5 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.

15.274.2.6 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

15.275 src/l2_packet/l2_packet_winpcap.c File Reference

WPA Supplicant - Layer2 packet handling with WinPcap RX thread. `#include "includes.h"`

```
#include <pcap.h>
#include "common.h"
#include "eloop.h"
#include "l2_packet.h"
```

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.

- struct 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.

15.275.1 Detailed Description

WPA Supplicant - Layer2 packet handling with WinPcap RX thread.

Copyright

Copyright (c) 2003-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This l2_packet implementation is explicitly for WinPcap and Windows events. l2_packet_pcap.c has support for WinPcap, but it requires polling to receive frames which means relatively long latency for EAPOL RX processing. The implementation here uses a separate thread to allow WinPcap to be receiving all the time to reduce latency for EAPOL receiving from about 100 ms to 3 ms when comparing l2_packet_pcap.c to l2_packet_winpcap.c. Extra sleep of 50 ms is added in to receive thread whenever no EAPOL frames has been received for a while. Whenever an EAPOL handshake is expected, this sleep is removed.

The RX thread receives a frame and signals main thread through Windows event about the availability of a new frame. Processing the received frame is synchronized with pair of Windows events so that no extra buffer or queuing mechanism is needed. This implementation requires Windows specific event loop implementation, i.e., eloop_win.c.

WinPcap has pcap_getevent() that could, in theory at least, be used to implement this kind of waiting with a simpler single-thread design. However, that event handle is not really signaled immediately when receiving each frame, so it does not really work for this kind of use.

15.275.2 Function Documentation

15.275.2.1 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()

15.275.2.2 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.

15.275.2.3 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

15.275.2.4 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*) [read]

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.

15.275.2.5 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.

15.275.2.6 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

15.276 src/radius/radius.c File Reference

hostapd / RADIUS message processing `#include "includes.h"`

```
#include "common.h"
#include "radius.h"
#include "md5.h"
#include "crypto.h"
```

Data Structures

- struct radius_attr_type
- struct radius_tunnel_attrs

Defines

- #define **RADIUS_ATTRS** (sizeof(radius_attrs) / sizeof(radius_attrs[0]))

Functions

- struct 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, const 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, const u8 *secret, size_t secret_len)
- struct radius_attr_hdr * **radius_msg_add_attr** (struct radius_msg *msg, u8 type, const u8 *data, size_t data_len)
- struct radius_msg * **radius_msg_parse** (const u8 *data, size_t len)
- int **radius_msg_add_eap** (struct radius_msg *msg, const u8 *data, size_t data_len)
- u8 * **radius_msg_get_eap** (struct radius_msg *msg, size_t *eap_len)
- int **radius_msg_verify_msg_auth** (struct radius_msg *msg, const u8 *secret, size_t secret_len, const u8 *req_auth)
- int **radius_msg_verify** (struct radius_msg *msg, const u8 *secret, size_t secret_len, struct radius_msg *sent_msg, int auth)
- int **radius_msg_copy_attr** (struct radius_msg *dst, struct radius_msg *src, u8 type)
- void **radius_msg_make_authenticator** (struct radius_msg *msg, const u8 *data, size_t len)
- struct radius_ms_mppe_keys * **radius_msg_get_ms_keys** (struct radius_msg *msg, struct radius_msg *sent_msg, const u8 *secret, size_t secret_len)
- struct radius_ms_mppe_keys * **radius_msg_get_cisco_keys** (struct radius_msg *msg, struct radius_msg *sent_msg, const 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)
- struct radius_attr_hdr * **radius_msg_add_attr_user_password** (struct radius_msg *msg, const u8 *data, size_t data_len, const u8 *secret, size_t secret_len)

- int **radius_msg_get_attr** (struct radius_msg *msg, u8 type, u8 *buf, size_t len)
- int **radius_msg_get_attr_ptr** (struct radius_msg *msg, u8 type, u8 **buf, size_t *len, const u8 *start)
- int **radius_msg_count_attr** (struct radius_msg *msg, u8 type, int min_len)
- int radius_msg_get_vlanid (struct radius_msg *msg)

  Parse RADIUS attributes for VLAN tunnel information.

- void **radius_free_class** (struct radius_class_data *c)
- int **radius_copy_class** (struct radius_class_data *dst, const struct radius_class_data *src)

15.276.1 Detailed Description

hostapd / RADIUS message processing

Copyright

Copyright (c) 2002-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.276.2 Function Documentation

15.276.2.1 int radius_msg_get_vlanid (struct radius_msg * *msg*)

Parse RADIUS attributes for VLAN tunnel information.

Parameters:

msg RADIUS message

Returns:

VLAN ID for the first tunnel configuration of -1 if none is found

15.277 src/radius/radius.h File Reference

hostapd / RADIUS message processing

Data Structures

- struct radius_hdr
- struct radius_attr_hdr
- struct radius_attr_vendor
- struct radius_ms_mppe_keys
- struct radius_msg
- struct radius_attr_data
- struct radius_class_data

Defines

- #define **RADIUS_MAX_ATTR_LEN** (255 - sizeof(struct radius_attr_hdr))
- #define **RADIUS_TERMINATION_ACTION_DEFAULT** 0
- #define **RADIUS_TERMINATION_ACTION_RADIUS_REQUEST** 1
- #define **RADIUS_NAS_PORT_TYPE_IEEE_802_11** 19
- #define **RADIUS_ACCT_STATUS_TYPE_START** 1
- #define **RADIUS_ACCT_STATUS_TYPE_STOP** 2
- #define **RADIUS_ACCT_STATUS_TYPE_INTERIM_UPDATE** 3
- #define **RADIUS_ACCT_STATUS_TYPE_ACCOUNTING_ON** 7
- #define **RADIUS_ACCT_STATUS_TYPE_ACCOUNTING_OFF** 8
- #define **RADIUS_ACCT_AUTHENTIC_RADIUS** 1
- #define **RADIUS_ACCT_AUTHENTIC_LOCAL** 2
- #define **RADIUS_ACCT_AUTHENTIC_REMOTE** 3
- #define **RADIUS_ACCT_TERMINATE_CAUSE_USER_REQUEST** 1
- #define **RADIUS_ACCT_TERMINATE_CAUSE_LOST_CARRIER** 2
- #define **RADIUS_ACCT_TERMINATE_CAUSE_LOST_SERVICE** 3
- #define **RADIUS_ACCT_TERMINATE_CAUSE_IDLE_TIMEOUT** 4
- #define **RADIUS_ACCT_TERMINATE_CAUSE_SESSION_TIMEOUT** 5
- #define **RADIUS_ACCT_TERMINATE_CAUSE_ADMIN_RESET** 6
- #define **RADIUS_ACCT_TERMINATE_CAUSE_ADMIN_REBOOT** 7
- #define **RADIUS_ACCT_TERMINATE_CAUSE_PORT_ERROR** 8
- #define **RADIUS_ACCT_TERMINATE_CAUSE_NAS_ERROR** 9
- #define **RADIUS_ACCT_TERMINATE_CAUSE_NAS_REQUEST** 10
- #define **RADIUS_ACCT_TERMINATE_CAUSE_NAS_REBOOT** 11
- #define **RADIUS_ACCT_TERMINATE_CAUSE_PORT_UNNEEDED** 12
- #define **RADIUS_ACCT_TERMINATE_CAUSE_PORT_PREEMPTED** 13
- #define **RADIUS_ACCT_TERMINATE_CAUSE_PORT_SUSPENDED** 14
- #define **RADIUS_ACCT_TERMINATE_CAUSE_SERVICE_UNAVAILABLE** 15
- #define **RADIUS_ACCT_TERMINATE_CAUSE_CALLBACK** 16
- #define **RADIUS_ACCT_TERMINATE_CAUSE_USER_ERROR** 17
- #define **RADIUS_ACCT_TERMINATE_CAUSE_HOST_REQUEST** 18
- #define **RADIUS_TUNNEL_TAGS** 32
- #define **RADIUS_TUNNEL_TYPE_PPTP** 1
- #define **RADIUS_TUNNEL_TYPE_L2TP** 3

- #define **RADIUS_TUNNEL_TYPE_IPIP** 7
- #define **RADIUS_TUNNEL_TYPE_GRE** 10
- #define **RADIUS_TUNNEL_TYPE_VLAN** 13
- #define **RADIUS_TUNNEL_MEDIUM_TYPE_IPV4** 1
- #define **RADIUS_TUNNEL_MEDIUM_TYPE_IPV6** 2
- #define **RADIUS_TUNNEL_MEDIUM_TYPE_802** 6
- #define **RADIUS_VENDOR_ID_CISCO** 9
- #define **RADIUS_CISCO_AV_PAIR** 1
- #define **RADIUS_VENDOR_ID_MICROSOFT** 311
- #define **RADIUS_DEFAULT_MSG_SIZE** 1024
- #define **RADIUS_DEFAULT_ATTR_COUNT** 16
- #define **RADIUS_802_1X_ADDR_FORMAT** "%02X-%02X-%02X-%02X-%02X-%02X"
- #define **RADIUS_ADDR_FORMAT** "%02x%02x%02x%02x%02x%02x"

Enumerations

- enum {

  RADIUS_CODE_ACCESS_REQUEST = 1, **RADIUS_CODE_ACCESS_ACCEPT** = 2, **RADIUS_CODE_ACCESS_REJECT** = 3, **RADIUS_CODE_ACCOUNTING_REQUEST** = 4,

  RADIUS_CODE_ACCOUNTING_RESPONSE = 5, **RADIUS_CODE_ACCESS_CHALLENGE** = 11, **RADIUS_CODE_STATUS_SERVER** = 12, **RADIUS_CODE_STATUS_CLIENT** = 13,

  RADIUS_CODE_RESERVED = 255 }
- enum {

  RADIUS_ATTR_USER_NAME = 1, **RADIUS_ATTR_USER_PASSWORD** = 2, **RADIUS_ATTR_NAS_IP_ADDRESS** = 4, **RADIUS_ATTR_NAS_PORT** = 5,

  RADIUS_ATTR_FRAMED_MTU = 12, **RADIUS_ATTR_REPLY_MESSAGE** = 18, **RADIUS_ATTR_STATE** = 24, **RADIUS_ATTR_CLASS** = 25,

  RADIUS_ATTR_VENDOR_SPECIFIC = 26, **RADIUS_ATTR_SESSION_TIMEOUT** = 27, **RADIUS_ATTR_IDLE_TIMEOUT** = 28, **RADIUS_ATTR_TERMINATION_ACTION** = 29,

  RADIUS_ATTR_CALLED_STATION_ID = 30, **RADIUS_ATTR_CALLING_STATION_ID** = 31, **RADIUS_ATTR_NAS_IDENTIFIER** = 32, **RADIUS_ATTR_PROXY_STATE** = 33,

  RADIUS_ATTR_ACCT_STATUS_TYPE = 40, **RADIUS_ATTR_ACCT_DELAY_TIME** = 41, **RADIUS_ATTR_ACCT_INPUT_OCTETS** = 42, **RADIUS_ATTR_ACCT_OUTPUT_OCTETS** = 43,

  RADIUS_ATTR_ACCT_SESSION_ID = 44, **RADIUS_ATTR_ACCT_AUTHENTIC** = 45, **RADIUS_ATTR_ACCT_SESSION_TIME** = 46, **RADIUS_ATTR_ACCT_INPUT_PACKETS** = 47,

  RADIUS_ATTR_ACCT_OUTPUT_PACKETS = 48, **RADIUS_ATTR_ACCT_TERMINATE_CAUSE** = 49, **RADIUS_ATTR_ACCT_MULTI_SESSION_ID** = 50, **RADIUS_ATTR_ACCT_LINK_COUNT** = 51,

  RADIUS_ATTR_ACCT_INPUT_GIGAWORDS = 52, **RADIUS_ATTR_ACCT_OUTPUT_GIGAWORDS** = 53, **RADIUS_ATTR_EVENT_TIMESTAMP** = 55, **RADIUS_ATTR_NAS_PORT_TYPE** = 61,

  RADIUS_ATTR_TUNNEL_TYPE = 64, **RADIUS_ATTR_TUNNEL_MEDIUM_TYPE** = 65, **RADIUS_ATTR_CONNECT_INFO** = 77, **RADIUS_ATTR_EAP_MESSAGE** = 79,

RADIUS_ATTR_MESSAGE_AUTHENTICATOR = 80, **RADIUS_ATTR_TUNNEL_-PRIVATE_GROUP_ID** = 81, **RADIUS_ATTR_ACCT_INTERIM_INTERVAL** = 85, **RADIUS_ATTR_CHARGEABLE_USER_IDENTITY** = 89,

RADIUS_ATTR_NAS_IPV6_ADDRESS = 95 }

- enum { **RADIUS_VENDOR_ATTR_MS_MPPE_SEND_KEY** = 16, **RADIUS_VENDOR_-ATTR_MS_MPPE_RECV_KEY** = 17 }

Functions

- struct 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, const 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, const u8 *secret, size_t secret_len)
- struct radius_attr_hdr * **radius_msg_add_attr** (struct radius_msg *msg, u8 type, const u8 *data, size_t data_len)
- struct radius_msg * **radius_msg_parse** (const u8 *data, size_t len)
- int **radius_msg_add_eap** (struct radius_msg *msg, const u8 *data, size_t data_len)
- u8 * **radius_msg_get_eap** (struct radius_msg *msg, size_t *len)
- int **radius_msg_verify** (struct radius_msg *msg, const u8 *secret, size_t secret_len, struct radius_-msg *sent_msg, int auth)
- int **radius_msg_verify_msg_auth** (struct radius_msg *msg, const u8 *secret, size_t secret_len, const u8 *req_auth)
- int **radius_msg_copy_attr** (struct radius_msg *dst, struct radius_msg *src, u8 type)
- void **radius_msg_make_authenticator** (struct radius_msg *msg, const u8 *data, size_t len)
- struct radius_ms_mppe_keys * **radius_msg_get_ms_keys** (struct radius_msg *msg, struct radius_-msg *sent_msg, const u8 *secret, size_t secret_len)
- struct radius_ms_mppe_keys * **radius_msg_get_cisco_keys** (struct radius_msg *msg, struct radius_msg *sent_msg, const 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)
- struct radius_attr_hdr * **radius_msg_add_attr_user_password** (struct radius_msg *msg, const u8 *data, size_t data_len, const u8 *secret, size_t secret_len)
- int **radius_msg_get_attr** (struct radius_msg *msg, u8 type, u8 *buf, size_t len)
- int radius_msg_get_vlanid (struct radius_msg *msg)

  Parse RADIUS attributes for VLAN tunnel information.

- int **radius_msg_get_attr_ptr** (struct radius_msg *msg, u8 type, u8 **buf, size_t *len, const u8 *start)
- int **radius_msg_count_attr** (struct radius_msg *msg, u8 type, int min_len)
- void **radius_free_class** (struct radius_class_data *c)
- int **radius_copy_class** (struct radius_class_data *dst, const struct radius_class_data *src)

Variables

- struct radius_hdr **STRUCT_PACKED**

15.277.1 Detailed Description

hostapd / RADIUS message processing

Copyright

Copyright (c) 2002-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.277.2 Function Documentation

15.277.2.1 int radius_msg_get_vlanid (struct radius_msg * *msg*)

Parse RADIUS attributes for VLAN tunnel information.

Parameters:

msg RADIUS message

Returns:

VLAN ID for the first tunnel configuration of -1 if none is found

15.278 src/radius/radius_client.c File Reference

RADIUS client. `#include "includes.h"`

`#include "common.h"`

`#include "radius.h"`

`#include "radius_client.h"`

`#include "eloop.h"`

Data Structures

- struct radius_rx_handler

  RADIUS client RX handler.

- struct radius_msg_list

  RADIUS client message retransmit list.

- struct radius_client_data

  Internal RADIUS client data.

Defines

- #define RADIUS_CLIENT_FIRST_WAIT 3

  RADIUS client timeout for first retry in seconds.

- #define RADIUS_CLIENT_MAX_WAIT 120

  RADIUS client maximum retry timeout in seconds.

- #define RADIUS_CLIENT_MAX_RETRIES 10

  RADIUS client maximum retries.

- #define RADIUS_CLIENT_MAX_ENTRIES 30

  RADIUS client maximum pending messages.

- #define RADIUS_CLIENT_NUM_FAILOVER 4

  RADIUS client failover point.

Functions

- int radius_client_register (struct radius_client_data *radius, RadiusType msg_type, RadiusRxResult(*handler)(struct radius_msg *msg, struct radius_msg *req, const u8 *shared_secret, size_t shared_secret_len, void *data), void *data)

  Register a RADIUS client RX handler.

- int radius_client_send (struct radius_client_data *radius, struct radius_msg *msg, RadiusType msg_type, const u8 *addr)

Send a RADIUS request.

- u8 radius_client_get_id (struct radius_client_data *radius)
  Get an identifier for a new RADIUS message.
- void radius_client_flush (struct radius_client_data *radius, int only_auth)
  Flush all pending RADIUS client messages.
- struct radius_client_data * radius_client_init (void *ctx, struct hostapd_radius_servers *conf)
  Initialize RADIUS client.
- void radius_client_deinit (struct radius_client_data *radius)
  Deinitialize RADIUS client.
- void radius_client_flush_auth (struct radius_client_data *radius, const u8 *addr)
  Flush pending RADIUS messages for an address.
- int radius_client_get_mib (struct radius_client_data *radius, char *buf, size_t buflen)
  Get RADIUS client MIB information.

15.278.1 Detailed Description

RADIUS client.

Copyright

Copyright (c) 2002-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.278.2 Define Documentation

15.278.2.1 #define RADIUS_CLIENT_MAX_ENTRIES 30

RADIUS client maximum pending messages. Maximum number of entries in retransmit list (oldest entries will be removed, if this limit is exceeded).

15.278.2.2 #define RADIUS_CLIENT_MAX_RETRIES 10

RADIUS client maximum retries. Maximum number of retransmit attempts before the entry is removed from retransmit list.

15.278.2.3 #define RADIUS_CLIENT_NUM_FAILOVER 4

RADIUS client failover point. The number of failed retry attempts after which the RADIUS server will be changed (if one of more backup servers are configured).

15.278.3 Function Documentation

15.278.3.1 void radius_client_deinit (struct radius_client_data * *radius*)

Deinitialize RADIUS client.

Parameters:

radius RADIUS client context from radius_client_init()

15.278.3.2 void radius_client_flush (struct radius_client_data * *radius*, int *only_auth*)

Flush all pending RADIUS client messages.

Parameters:

radius RADIUS client context from radius_client_init()

only_auth Whether only authentication messages are removed

15.278.3.3 void radius_client_flush_auth (struct radius_client_data * *radius*, const u8 * *addr*)

Flush pending RADIUS messages for an address.

Parameters:

radius RADIUS client context from radius_client_init()

addr MAC address of the related device

This function can be used to remove pending RADIUS authentication messages that are related to a specific device. The addr parameter is matched with the one used in radius_client_send() call that was used to transmit the authentication request.

15.278.3.4 u8 radius_client_get_id (struct radius_client_data * *radius*)

Get an identifier for a new RADIUS message.

Parameters:

radius RADIUS client context from radius_client_init()

Returns:

Allocated identifier

This function is used to fetch a unique (among pending requests) identifier for a new RADIUS message.

15.278.3.5 int radius_client_get_mib (struct radius_client_data * *radius*, char * *buf*, size_t *buflen*)

Get RADIUS client MIB information.

Parameters:

radius RADIUS client context from radius_client_init()

buf Buffer for returning MIB data in text format

buflen Maximum buf length in octets

Returns:

Number of octets written into the buffer

15.278.3.6 struct radius_client_data* radius_client_init (void * *ctx*, struct hostapd_radius_servers * *conf*) [read]

Initialize RADIUS client.

Parameters:

ctx Callback context to be used in hostapd_logger() calls

conf RADIUS client configuration (RADIUS servers)

Returns:

Pointer to private RADIUS client context or NULL on failure

The caller is responsible for keeping the configuration data available for the lifetime of the RADIUS client, i.e., until radius_client_deinit() is called for the returned context pointer.

15.278.3.7 int radius_client_register (struct radius_client_data * *radius*, RadiusType *msg_type*, RadiusRxResult(*)(struct radius_msg *msg, struct radius_msg *req, const u8 *shared_secret, size_t shared_secret_len, void *data) *handler*, void * *data*)

Register a RADIUS client RX handler.

Parameters:

radius RADIUS client context from radius_client_init()

msg_type RADIUS client type (RADIUS_AUTH or RADIUS_ACCT)

handler Handler for received RADIUS messages

data Context pointer for handler callbacks

Returns:

0 on success, -1 on failure

This function is used to register a handler for processing received RADIUS authentication and accounting messages. The handler() callback function will be called whenever a RADIUS message is received from the active server.

There can be multiple registered RADIUS message handlers. The handlers will be called in order until one of them indicates that it has processed or queued the message.

15.278.3.8 int radius_client_send (struct radius_client_data * *radius*, struct radius_msg * *msg*, RadiusType *msg_type*, const u8 * *addr*)

Send a RADIUS request.

Parameters:

radius RADIUS client context from radius_client_init()

msg RADIUS message to be sent

msg_type Message type (RADIUS_AUTH, RADIUS_ACCT, RADIUS_ACCT_INTERIM)

addr MAC address of the device related to this message or NULL

Returns:

0 on success, -1 on failure

This function is used to transmit a RADIUS authentication (RADIUS_AUTH) or accounting request (RADIUS_ACCT or RADIUS_ACCT_INTERIM). The only difference between accounting and interim accounting messages is that the interim message will override any pending interim accounting updates while a new accounting message does not remove any pending messages.

The message is added on the retransmission queue and will be retransmitted automatically until a response is received or maximum number of retries (RADIUS_CLIENT_MAX_RETRIES) is reached.

The related device MAC address can be used to identify pending messages that can be removed with radius_client_flush_auth() or with interim accounting updates.

15.279 src/radius/radius_client.h File Reference

RADIUS client. `#include "ip_addr.h"`

Data Structures

- struct hostapd_radius_server

  RADIUS server information for RADIUS client.

- struct hostapd_radius_servers

  RADIUS servers for RADIUS client.

Enumerations

- enum RadiusType { RADIUS_AUTH, RADIUS_ACCT, RADIUS_ACCT_INTERIM }

  RADIUS server type for RADIUS client.

- enum RadiusRxResult { **RADIUS_RX_PROCESSED**, **RADIUS_RX_QUEUED**, **RADIUS_RX_UNKNOWN**, **RADIUS_RX_INVALID_AUTHENTICATOR** }

  RADIUS client RX handler result.

Functions

- int radius_client_register (struct radius_client_data *radius, RadiusType msg_type, RadiusRxResult(*handler)(struct radius_msg *msg, struct radius_msg *req, const u8 *shared_secret, size_t shared_secret_len, void *data), void *data)

  Register a RADIUS client RX handler.

- int radius_client_send (struct radius_client_data *radius, struct radius_msg *msg, RadiusType msg_type, const u8 *addr)

  Send a RADIUS request.

- u8 radius_client_get_id (struct radius_client_data *radius)

  Get an identifier for a new RADIUS message.

- void radius_client_flush (struct radius_client_data *radius, int only_auth)

  Flush all pending RADIUS client messages.

- struct radius_client_data * radius_client_init (void *ctx, struct hostapd_radius_servers *conf)

  Initialize RADIUS client.

- void radius_client_deinit (struct radius_client_data *radius)

  Deinitialize RADIUS client.

- void radius_client_flush_auth (struct radius_client_data *radius, const u8 *addr)

  Flush pending RADIUS messages for an address.

- int radius_client_get_mib (struct radius_client_data *radius, char *buf, size_t buflen)
  Get RADIUS client MIB information.

15.279.1 Detailed Description

RADIUS client.

Copyright

Copyright (c) 2002-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.279.2 Enumeration Type Documentation

15.279.2.1 enum RadiusType

RADIUS server type for RADIUS client.

Enumerator:

RADIUS_AUTH RADIUS authentication

RADIUS_ACCT RADIUS accounting.

RADIUS_ACCT_INTERIM RADIUS interim accounting message. Used only with radius_client_send(). This behaves just like RADIUS_ACCT, but removes any pending interim RADIUS Accounting messages for the same STA before sending the new interim update.

15.279.3 Function Documentation

15.279.3.1 void radius_client_deinit (struct radius_client_data * *radius*)

Deinitialize RADIUS client.

Parameters:

radius RADIUS client context from radius_client_init()

15.279.3.2 void radius_client_flush (struct radius_client_data * *radius*, int *only_auth*)

Flush all pending RADIUS client messages.

Parameters:

radius RADIUS client context from radius_client_init()

only_auth Whether only authentication messages are removed

15.279.3.3 void radius_client_flush_auth (struct radius_client_data * *radius*, const u8 * *addr*)

Flush pending RADIUS messages for an address.

Parameters:

radius RADIUS client context from radius_client_init()

addr MAC address of the related device

This function can be used to remove pending RADIUS authentication messages that are related to a specific device. The addr parameter is matched with the one used in radius_client_send() call that was used to transmit the authentication request.

15.279.3.4 u8 radius_client_get_id (struct radius_client_data * *radius*)

Get an identifier for a new RADIUS message.

Parameters:

radius RADIUS client context from radius_client_init()

Returns:

Allocated identifier

This function is used to fetch a unique (among pending requests) identifier for a new RADIUS message.

15.279.3.5 int radius_client_get_mib (struct radius_client_data * *radius*, char * *buf*, size_t *buflen*)

Get RADIUS client MIB information.

Parameters:

radius RADIUS client context from radius_client_init()

buf Buffer for returning MIB data in text format

buflen Maximum buf length in octets

Returns:

Number of octets written into the buffer

15.279.3.6 struct radius_client_data* radius_client_init (void * *ctx*, struct hostapd_radius_servers * *conf*) [read]

Initialize RADIUS client.

Parameters:

ctx Callback context to be used in hostapd_logger() calls

conf RADIUS client configuration (RADIUS servers)

Returns:

Pointer to private RADIUS client context or NULL on failure

The caller is responsible for keeping the configuration data available for the lifetime of the RADIUS client, i.e., until radius_client_deinit() is called for the returned context pointer.

15.279.3.7 int radius_client_register (struct radius_client_data * *radius*, RadiusType *msg_type*, RadiusRxResult(*)(struct radius_msg *msg, struct radius_msg *req, const u8 *shared_secret, size_t shared_secret_len, void *data) *handler*, void * *data*)

Register a RADIUS client RX handler.

Parameters:

radius RADIUS client context from radius_client_init()

msg_type RADIUS client type (RADIUS_AUTH or RADIUS_ACCT)

handler Handler for received RADIUS messages

data Context pointer for handler callbacks

Returns:

0 on success, -1 on failure

This function is used to register a handler for processing received RADIUS authentication and accounting messages. The handler() callback function will be called whenever a RADIUS message is received from the active server.

There can be multiple registered RADIUS message handlers. The handlers will be called in order until one of them indicates that it has processed or queued the message.

15.279.3.8 int radius_client_send (struct radius_client_data * *radius*, struct radius_msg * *msg*, RadiusType *msg_type*, const u8 * *addr*)

Send a RADIUS request.

Parameters:

radius RADIUS client context from radius_client_init()

msg RADIUS message to be sent

msg_type Message type (RADIUS_AUTH, RADIUS_ACCT, RADIUS_ACCT_INTERIM)

addr MAC address of the device related to this message or NULL

Returns:

0 on success, -1 on failure

This function is used to transmit a RADIUS authentication (RADIUS_AUTH) or accounting request (RADIUS_ACCT or RADIUS_ACCT_INTERIM). The only difference between accounting and interim accounting messages is that the interim message will override any pending interim accounting updates while a new accounting message does not remove any pending messages.

The message is added on the retransmission queue and will be retransmitted automatically until a response is received or maximum number of retries (RADIUS_CLIENT_MAX_RETRIES) is reached.

The related device MAC address can be used to identify pending messages that can be removed with radius_client_flush_auth() or with interim accounting updates.

15.280 src/radius/radius_server.c File Reference

hostapd / RADIUS authentication server `#include "includes.h"`

```
#include <net/if.h>
#include "common.h"
#include "radius.h"
#include "eloop.h"
#include "defs.h"
#include "eap_server/eap.h"
#include "radius_server.h"
```

Data Structures

- struct radius_server_counters
- struct radius_session
- struct radius_client
- struct radius_server_data

Defines

- #define **RADIUS_SESSION_TIMEOUT** 60
- #define **RADIUS_MAX_SESSION** 100
- #define **RADIUS_MAX_MSG_LEN** 3000
- #define **RADIUS_DEBUG**(args...) wpa_printf(MSG_DEBUG, "RADIUS SRV: " args)
- #define **RADIUS_ERROR**(args...) wpa_printf(MSG_ERROR, "RADIUS SRV: " args)
- #define **RADIUS_DUMP**(args...) wpa_hexdump(MSG_MSGDUMP, "RADIUS SRV: " args)
- #define **RADIUS_DUMP_ASCII**(args...) wpa_hexdump_ascii(MSG_MSGDUMP, "RADIUS SRV: " args)

Functions

- struct radius_server_data * **radius_server_init** (struct radius_server_conf *conf)
- void **radius_server_deinit** (struct radius_server_data *data)
- int **radius_server_get_mib** (struct radius_server_data *data, char *buf, size_t buflen)
- void **radius_server_eap_pending_cb** (struct radius_server_data *data, void *ctx)

Variables

- int **wpa_debug_level**

15.280.1 Detailed Description

hostapd / RADIUS authentication server

Copyright

15.281 src/radius/radius_server.h File Reference

hostapd / RADIUS authentication server

Data Structures

- struct radius_server_conf

15.281.1 Detailed Description

hostapd / RADIUS authentication server

Copyright

15.282 src/rsn_supp/peerkey.h File Reference

WPA Supplicant - PeerKey for Direct Link Setup (DLS).

Data Structures

- struct wpa_peerkey

Defines

- #define **PEERKEY_MAX_IE_LEN** 80

15.282.1 Detailed Description

WPA Supplicant - PeerKey for Direct Link Setup (DLS).

Copyright

15.283 src/rsn_supp/wpa_i.h File Reference

wpa_supplicant - Internal WPA state machine definitions

Data Structures

- struct wpa_sm

  Internal WPA state machine data.

Functions

- void wpa_eapol_key_send (struct wpa_sm ∗sm, const u8 ∗kck, int ver, const u8 ∗dest, u16 proto, u8 ∗msg, size_t msg_len, u8 ∗key_mic)

  Send WPA/RSN EAPOL-Key message.

- int wpa_supplicant_send_2_of_4 (struct wpa_sm ∗sm, const unsigned char ∗dst, const struct wpa_eapol_key ∗key, int ver, const u8 ∗nonce, const u8 ∗wpa_ie, size_t wpa_ie_len, struct wpa_ptk ∗ptk)

  Send message 2 of WPA/RSN 4-Way Handshake.

- int wpa_supplicant_send_4_of_4 (struct wpa_sm ∗sm, const unsigned char ∗dst, const struct wpa_eapol_key ∗key, u16 ver, u16 key_info, const u8 ∗kde, size_t kde_len, struct wpa_ptk ∗ptk)

  Send message 4 of WPA/RSN 4-Way Handshake.

- int **wpa_derive_ptk_ft** (struct wpa_sm ∗sm, const unsigned char ∗src_addr, const struct wpa_eapol_key ∗key, struct wpa_ptk ∗ptk, size_t ptk_len)

15.283.1 Detailed Description

wpa_supplicant - Internal WPA state machine definitions

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.283.2 Function Documentation

15.283.2.1 void wpa_eapol_key_send (struct wpa_sm ∗ *sm*, const u8 ∗ *kck*, int *ver*, const u8 ∗ *dest*, u16 *proto*, u8 ∗ *msg*, size_t *msg_len*, u8 ∗ *key_mic*)

Send WPA/RSN EAPOL-Key message.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

kck Key Confirmation Key (KCK, part of PTK)

ver Version field from Key Info

dest Destination address for the frame

proto Ethertype (usually ETH_P_EAPOL)

msg EAPOL-Key message

msg_len Length of message

key_mic Pointer to the buffer to which the EAPOL-Key MIC is written

15.283.2.2 int wpa_supplicant_send_2_of_4 (struct wpa_sm * *sm*, const unsigned char * *dst*, const struct wpa_eapol_key * *key*, int *ver*, const u8 * *nonce*, const u8 * *wpa_ie*, size_t *wpa_ie_len*, struct wpa_ptk * *ptk*)

Send message 2 of WPA/RSN 4-Way Handshake.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

dst Destination address for the frame

key Pointer to the EAPOL-Key frame header

ver Version bits from EAPOL-Key Key Info

nonce Nonce value for the EAPOL-Key frame

wpa_ie WPA/RSN IE

wpa_ie_len Length of the WPA/RSN IE

ptk PTK to use for keyed hash and encryption

Returns:

0 on success, -1 on failure

15.283.2.3 int wpa_supplicant_send_4_of_4 (struct wpa_sm * *sm*, const unsigned char * *dst*, const struct wpa_eapol_key * *key*, u16 *ver*, u16 *key_info*, const u8 * *kde*, size_t *kde_len*, struct wpa_ptk * *ptk*)

Send message 4 of WPA/RSN 4-Way Handshake.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

dst Destination address for the frame

key Pointer to the EAPOL-Key frame header

ver Version bits from EAPOL-Key Key Info

key_info Key Info

kde KDEs to include the EAPOL-Key frame

kde_len Length of KDEs

ptk PTK to use for keyed hash and encryption

Returns:

0 on success, -1 on failure

15.284 src/rsn_supp/wpa_ie.c File Reference

wpa_supplicant - WPA/RSN IE and KDE processing `#include "includes.h"`

```
#include "common.h"
#include "wpa.h"
#include "pmksa_cache.h"
#include "ieee802_11_defs.h"
#include "wpa_i.h"
#include "wpa_ie.h"
```

Functions

- int wpa_parse_wpa_ie (const u8 *wpa_ie, size_t wpa_ie_len, struct wpa_ie_data *data)
  Parse WPA/RSN IE.
- int wpa_gen_wpa_ie (struct wpa_sm *sm, u8 *wpa_ie, size_t wpa_ie_len)
  Generate WPA/RSN IE based on current security policy.
- int wpa_supplicant_parse_ies (const u8 *buf, size_t len, struct wpa_eapol_ie_parse *ie)
  Parse EAPOL-Key Key Data IEs.

15.284.1 Detailed Description

wpa_supplicant - WPA/RSN IE and KDE processing

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.284.2 Function Documentation

15.284.2.1 int wpa_gen_wpa_ie (struct wpa_sm * *sm*, u8 * *wpa_ie*, size_t *wpa_ie_len*)

Generate WPA/RSN IE based on current security policy.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

wpa_ie Pointer to memory area for the generated WPA/RSN IE

wpa_ie_len Maximum length of the generated WPA/RSN IE

Returns:

Length of the generated WPA/RSN IE or -1 on failure

15.284.2.2 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.

15.284.2.3 int wpa_supplicant_parse_ies (const u8 * *buf*, size_t *len*, struct wpa_eapol_ie_parse * *ie*)

Parse EAPOL-Key Key Data IEs.

Parameters:

buf Pointer to the Key Data buffer

len Key Data Length

ie Pointer to parsed IE data

Returns:

0 on success, -1 on failure

15.285 src/rsn_supp/wpa_ie.h File Reference

wpa_supplicant - WPA/RSN IE and KDE definitions

Data Structures

- struct wpa_eapol_ie_parse

Functions

- int wpa_supplicant_parse_ies (const u8 *buf, size_t len, struct wpa_eapol_ie_parse *ie)

  Parse EAPOL-Key Key Data IEs.

- int wpa_gen_wpa_ie (struct wpa_sm *sm, u8 *wpa_ie, size_t wpa_ie_len)

  Generate WPA/RSN IE based on current security policy.

15.285.1 Detailed Description

wpa_supplicant - WPA/RSN IE and KDE definitions

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.285.2 Function Documentation

15.285.2.1 int wpa_gen_wpa_ie (struct wpa_sm * *sm*, u8 * *wpa_ie*, size_t *wpa_ie_len*)

Generate WPA/RSN IE based on current security policy.

Parameters:

sm Pointer to WPA state machine data from wpa_sm_init()

wpa_ie Pointer to memory area for the generated WPA/RSN IE

wpa_ie_len Maximum length of the generated WPA/RSN IE

Returns:

Length of the generated WPA/RSN IE or -1 on failure

15.285.2.2 int wpa_supplicant_parse_ies (const u8 * *buf*, size_t *len*, struct wpa_eapol_ie_parse * *ie*)

Parse EAPOL-Key Key Data IEs.

Parameters:

buf Pointer to the Key Data buffer

len Key Data Length

ie Pointer to parsed IE data

Returns:

0 on success, -1 on failure

15.286 src/tls/asn1.c File Reference

ASN.1 DER parsing. `#include "includes.h"`

`#include "common.h"`

15.286.1 Detailed Description

ASN.1 DER parsing.

Copyright

Copyright (c) 2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.287 src/tls/asn1.h File Reference

ASN.1 DER parsing.

Data Structures

- struct asn1_hdr
- struct asn1_oid

Defines

- #define **ASN1_TAG_EOC** 0x00
- #define **ASN1_TAG_BOOLEAN** 0x01
- #define **ASN1_TAG_INTEGER** 0x02
- #define **ASN1_TAG_BITSTRING** 0x03
- #define **ASN1_TAG_OCTETSTRING** 0x04
- #define **ASN1_TAG_NULL** 0x05
- #define **ASN1_TAG_OID** 0x06
- #define **ASN1_TAG_OBJECT_DESCRIPTOR** 0x07
- #define **ASN1_TAG_EXTERNAL** 0x08
- #define **ASN1_TAG_REAL** 0x09
- #define **ASN1_TAG_ENUMERATED** 0x0A
- #define **ASN1_TAG_UTF8STRING** 0x0C
- #define **ANS1_TAG_RELATIVE_OID** 0x0D
- #define **ASN1_TAG_SEQUENCE** 0x10
- #define **ASN1_TAG_SET** 0x11
- #define **ASN1_TAG_NUMERICSTRING** 0x12
- #define **ASN1_TAG_PRINTABLESTRING** 0x13
- #define **ASN1_TAG_TG1STRING** 0x14
- #define **ASN1_TAG_VIDEOTEXSTRING** 0x15
- #define **ASN1_TAG_IA5STRING** 0x16
- #define **ASN1_TAG_UTCTIME** 0x17
- #define **ASN1_TAG_GENERALIZEDTIME** 0x18
- #define **ASN1_TAG_GRAPHICSTRING** 0x19
- #define **ASN1_TAG_VISIBLESTRING** 0x1A
- #define **ASN1_TAG_GENERALSTRING** 0x1B
- #define **ASN1_TAG_UNIVERSALSTRING** 0x1C
- #define **ASN1_TAG_BMPSTRING** 0x1D
- #define **ASN1_CLASS_UNIVERSAL** 0
- #define **ASN1_CLASS_APPLICATION** 1
- #define **ASN1_CLASS_CONTEXT_SPECIFIC** 2
- #define **ASN1_CLASS_PRIVATE** 3
- #define **ASN1_MAX_OID_LEN** 20

Functions

- int **asn1_get_next** (const u8 *buf, size_t len, struct asn1_hdr *hdr)
- int **asn1_parse_oid** (const u8 *buf, size_t len, struct asn1_oid *oid)
- int **asn1_get_oid** (const u8 *buf, size_t len, struct asn1_oid *oid, const u8 **next)
- void **asn1_oid_to_str** (struct asn1_oid *oid, char *buf, size_t len)
- unsigned long **asn1_bit_string_to_long** (const u8 *buf, size_t len)

15.287.1 Detailed Description

ASN.1 DER parsing.

Copyright

15.288 src/tls/asn1_test.c File Reference

Testing tool for ASN.1/X.509v3 routines. `#include "includes.h"`

`#include "common.h"`

`#include "asn1.h"`

`#include "x509v3.h"`

Functions

- int **asn1_parse** (const u8 *buf, size_t len, int level)
- int **main** (int argc, char *argv[ ])

Variables

- int **wpa_debug_level**

15.288.1 Detailed Description

Testing tool for ASN.1/X.509v3 routines.

Copyright

Copyright (c) 2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.289 src/tls/bignum.c File Reference

Big number math. `#include "includes.h"`

`#include "common.h"`

`#include "bignum.h"`

`#include <tommath.h>`

Functions

- struct bignum * bignum_init (void)

  Allocate memory for bignum.

- void bignum_deinit (struct bignum *n)

  Free bignum.

- size_t bignum_get_unsigned_bin_len (struct bignum *n)

  Get length of bignum as an unsigned binary buffer.

- int bignum_get_unsigned_bin (const struct bignum *n, u8 *buf, size_t *len)

  Set binary buffer to unsigned bignum.

- int bignum_set_unsigned_bin (struct bignum *n, const u8 *buf, size_t len)

  Set bignum based on unsigned binary buffer.

- int bignum_cmp (const struct bignum *a, const struct bignum *b)

  Signed comparison.

- int bignum_cmp_d (const struct bignum *a, unsigned long b)

  Compare bignum to standard integer.

- int bignum_add (const struct bignum *a, const struct bignum *b, struct bignum *c)

  c = a + b

- int bignum_sub (const struct bignum *a, const struct bignum *b, struct bignum *c)

  c = a - b

- int bignum_mul (const struct bignum *a, const struct bignum *b, struct bignum *c)

  *c = a * b*

- int bignum_mulmod (const struct bignum *a, const struct bignum *b, const struct bignum *c, struct bignum *d)

  *d = a * b (mod c)*

- int bignum_exptmod (const struct bignum *a, const struct bignum *b, const struct bignum *c, struct bignum *d)

  Modular exponentiation: d = a^b (mod c).

15.289.1 Detailed Description

Big number math.

Copyright

Copyright (c) 2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.289.2 Function Documentation

15.289.2.1 int bignum_add (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a + b

Parameters:

- ***a*** Bignum from bignum_init()
- ***b*** Bignum from bignum_init()
- ***c*** Bignum from bignum_init(); used to store the result of a + b

Returns:

0 on success, -1 on failure

15.289.2.2 int bignum_cmp (const struct bignum * *a*, const struct bignum * *b*)

Signed comparison.

Parameters:

- ***a*** Bignum from bignum_init()
- ***b*** Bignum from bignum_init()

Returns:

0 on success, -1 on failure

15.289.2.3 int bignum_cmp_d (const struct bignum * *a*, unsigned long *b*)

Compare bignum to standard integer.

Parameters:

- ***a*** Bignum from bignum_init()
- ***b*** Small integer

Returns:

0 on success, -1 on failure

15.289.2.4 void bignum_deinit (struct bignum * *n*)

Free bignum.

Parameters:

n Bignum from bignum_init()

15.289.2.5 int bignum_exptmod (const struct bignum * *a*, const struct bignum * *b*, const struct bignum * *c*, struct bignum * *d*)

Modular exponentiation: d = a^b (mod c).

Parameters:

a Bignum from bignum_init(); base

b Bignum from bignum_init(); exponent

c Bignum from bignum_init(); modulus

d Bignum from bignum_init(); used to store the result of a^b (mod c)

Returns:

0 on success, -1 on failure

15.289.2.6 int bignum_get_unsigned_bin (const struct bignum * *n*, u8 * *buf*, size_t * *len*)

Set binary buffer to unsigned bignum.

Parameters:

n Bignum from bignum_init()

buf Buffer for the binary number

len Length of the buffer, can be NULL if buffer is known to be long enough. Set to used buffer length on success if not NULL.

Returns:

0 on success, -1 on failure

15.289.2.7 size_t bignum_get_unsigned_bin_len (struct bignum * *n*)

Get length of bignum as an unsigned binary buffer.

Parameters:

n Bignum from bignum_init()

Returns:

Length of n if written to a binary buffer

15.289.2.8 struct bignum* bignum_init (void) [read]

Allocate memory for bignum.

Returns:

Pointer to allocated bignum or NULL on failure

15.289.2.9 int bignum_mul (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a * b

Parameters:

a Bignum from bignum_init()

b Bignum from bignum_init()

c Bignum from bignum_init(); used to store the result of a * b

Returns:

0 on success, -1 on failure

15.289.2.10 int bignum_mulmod (const struct bignum * *a*, const struct bignum * *b*, const struct bignum * *c*, struct bignum * *d*)

d = a * b (mod c)

Parameters:

a Bignum from bignum_init()

b Bignum from bignum_init()

c Bignum from bignum_init(); modulus

d Bignum from bignum_init(); used to store the result of a * b (mod c)

Returns:

0 on success, -1 on failure

15.289.2.11 int bignum_set_unsigned_bin (struct bignum * *n*, const u8 * *buf*, size_t *len*)

Set bignum based on unsigned binary buffer.

Parameters:

n Bignum from bignum_init(); to be set to the given value

buf Buffer with unsigned binary value

len Length of buf in octets

Returns:

0 on success, -1 on failure

15.289.2.12 int bignum_sub (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a - b

Parameters:

a Bignum from bignum_init()

b Bignum from bignum_init()

c Bignum from bignum_init(); used to store the result of a - b

Returns:

0 on success, -1 on failure

15.290 src/tls/bignum.h File Reference

Big number math.

Functions

- struct bignum * bignum_init (void)

  Allocate memory for bignum.

- void bignum_deinit (struct bignum *n)

  Free bignum.

- size_t bignum_get_unsigned_bin_len (struct bignum *n)

  Get length of bignum as an unsigned binary buffer.

- int bignum_get_unsigned_bin (const struct bignum *n, u8 *buf, size_t *len)

  Set binary buffer to unsigned bignum.

- int bignum_set_unsigned_bin (struct bignum *n, const u8 *buf, size_t len)

  Set bignum based on unsigned binary buffer.

- int bignum_cmp (const struct bignum *a, const struct bignum *b)

  Signed comparison.

- int bignum_cmp_d (const struct bignum *a, unsigned long b)

  Compare bignum to standard integer.

- int bignum_add (const struct bignum *a, const struct bignum *b, struct bignum *c)

  c = a + b

- int bignum_sub (const struct bignum *a, const struct bignum *b, struct bignum *c)

  c = a - b

- int bignum_mul (const struct bignum *a, const struct bignum *b, struct bignum *c)

  *c = a * b*

- int bignum_mulmod (const struct bignum *a, const struct bignum *b, const struct bignum *c, struct bignum *d)

  *d = a * b (mod c)*

- int bignum_exptmod (const struct bignum *a, const struct bignum *b, const struct bignum *c, struct bignum *d)

  Modular exponentiation: d = a^b (mod c).

15.290.1 Detailed Description

Big number math.

Copyright

Copyright (c) 2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.290.2 Function Documentation

15.290.2.1 int bignum_add (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a + b

Parameters:

- ***a*** Bignum from bignum_init()
- ***b*** Bignum from bignum_init()
- ***c*** Bignum from bignum_init(); used to store the result of a + b

Returns:

0 on success, -1 on failure

15.290.2.2 int bignum_cmp (const struct bignum * *a*, const struct bignum * *b*)

Signed comparison.

Parameters:

- ***a*** Bignum from bignum_init()
- ***b*** Bignum from bignum_init()

Returns:

0 on success, -1 on failure

15.290.2.3 int bignum_cmp_d (const struct bignum * *a*, unsigned long *b*)

Compare bignum to standard integer.

Parameters:

- ***a*** Bignum from bignum_init()
- ***b*** Small integer

Returns:

0 on success, -1 on failure

15.290.2.4 void bignum_deinit (struct bignum * *n*)

Free bignum.

Parameters:

n Bignum from bignum_init()

15.290.2.5 int bignum_exptmod (const struct bignum * *a*, const struct bignum * *b*, const struct bignum * *c*, struct bignum * *d*)

Modular exponentiation: d = a^b (mod c).

Parameters:

a Bignum from bignum_init(); base

b Bignum from bignum_init(); exponent

c Bignum from bignum_init(); modulus

d Bignum from bignum_init(); used to store the result of a^b (mod c)

Returns:

0 on success, -1 on failure

15.290.2.6 int bignum_get_unsigned_bin (const struct bignum * *n*, u8 * *buf*, size_t * *len*)

Set binary buffer to unsigned bignum.

Parameters:

n Bignum from bignum_init()

buf Buffer for the binary number

len Length of the buffer, can be NULL if buffer is known to be long enough. Set to used buffer length on success if not NULL.

Returns:

0 on success, -1 on failure

15.290.2.7 size_t bignum_get_unsigned_bin_len (struct bignum * *n*)

Get length of bignum as an unsigned binary buffer.

Parameters:

n Bignum from bignum_init()

Returns:

Length of n if written to a binary buffer

15.290.2.8 struct bignum* bignum_init (void) [read]

Allocate memory for bignum.

Returns:

Pointer to allocated bignum or NULL on failure

15.290.2.9 int bignum_mul (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a * b

Parameters:

a Bignum from bignum_init()

b Bignum from bignum_init()

c Bignum from bignum_init(); used to store the result of a * b

Returns:

0 on success, -1 on failure

15.290.2.10 int bignum_mulmod (const struct bignum * *a*, const struct bignum * *b*, const struct bignum * *c*, struct bignum * *d*)

d = a * b (mod c)

Parameters:

a Bignum from bignum_init()

b Bignum from bignum_init()

c Bignum from bignum_init(); modulus

d Bignum from bignum_init(); used to store the result of a * b (mod c)

Returns:

0 on success, -1 on failure

15.290.2.11 int bignum_set_unsigned_bin (struct bignum * *n*, const u8 * *buf*, size_t *len*)

Set bignum based on unsigned binary buffer.

Parameters:

n Bignum from bignum_init(); to be set to the given value

buf Buffer with unsigned binary value

len Length of buf in octets

Returns:

0 on success, -1 on failure

15.290.2.12 int bignum_sub (const struct bignum * *a*, const struct bignum * *b*, struct bignum * *c*)

c = a - b

Parameters:

a Bignum from bignum_init()

b Bignum from bignum_init()

c Bignum from bignum_init(); used to store the result of a - b

Returns:

0 on success, -1 on failure

15.291 src/tls/libtommath.c File Reference

Minimal code for RSA support from LibTomMath 0.41 http://libtom.org/ http://libtom.org/files/ltm-0.41.tar.bz2 This library was released in public domain by Tom St Denis.

Data Structures

- struct mp_int

Defines

- #define **CHAR_BIT** 8
- #define **BN_MP_INVMOD_C**
- #define **BN_S_MP_EXPTMOD_C**
- #define **BN_S_MP_MUL_DIGS_C**
- #define **BN_MP_INVMOD_SLOW_C**
- #define **BN_S_MP_SQR_C**
- #define **BN_S_MP_MUL_HIGH_DIGS_C**
- #define **BN_MP_DIV_SMALL**
- #define **BN_MP_INIT_MULTI_C**
- #define **BN_MP_CLEAR_MULTI_C**
- #define **BN_MP_ABS_C**
- #define **LTM_NO_NEG_EXP**
- #define **MAX**(x, y) ((x)>(y)?(x):(y))
- #define **OPT_CAST**(x)
- #define **DIGIT_BIT** 28
- #define **MP_28BIT**
- #define **XMALLOC** os_malloc
- #define **XFREE** os_free
- #define **XREALLOC** os_realloc
- #define **MP_MASK** ((((mp_digit)1)<<((mp_digit)DIGIT_BIT))-((mp_digit)1))
- #define **MP_LT** -1
- #define **MP_EQ** 0
- #define **MP_GT** 1
- #define **MP_ZPOS** 0
- #define **MP_NEG** 1
- #define **MP_OKAY** 0
- #define **MP_MEM** -2
- #define **MP_VAL** -3
- #define **MP_YES** 1
- #define **MP_NO** 0
- #define **MP_LOW_MEM**
- #define **MP_PREC** 8
- #define **MP_WARRAY** (1 << (sizeof(mp_word) * CHAR_BIT - 2 * DIGIT_BIT + 1))
- #define **mp_iszero**(a) (((a)->used == 0) ? MP_YES : MP_NO)
- #define **mp_iseven**(a) (((a)->used > 0 && (((a)->dp[0] & 1) == 0)) ? MP_YES : MP_NO)
- #define **mp_isodd**(a) (((a)->used > 0 && (((a)->dp[0] & 1) == 1)) ? MP_YES : MP_NO)
- #define **s_mp_mul**(a, b, c) s_mp_mul_digs(a, b, c, (a)->used + (b)->used + 1)
- #define **TAB_SIZE** 32

Typedefs

- typedef unsigned long **mp_digit**
- typedef u64 **mp_word**
- typedef int **mp_err**

15.291.1 Detailed Description

Minimal code for RSA support from LibTomMath 0.41 http://libtom.org/ http://libtom.org/files/ltm-0.41.tar.bz2 This library was released in public domain by Tom St Denis. The combination in this file may not use all of the optimized algorithms from LibTomMath and may be considerable slower than the LibTomMath with its default settings. The main purpose of having this version here is to make it easier to build bignum.c wrapper without having to install and build an external library.

If CONFIG_INTERNAL_LIBTOMMATH is defined, bignum.c includes this libtommath.c file instead of using the external LibTomMath library.

15.292 src/tls/pkcs1.c File Reference

PKCS #1 (RSA Encryption). `#include "includes.h"`

`#include "common.h"`

`#include "rsa.h"`

`#include "pkcs1.h"`

Functions

- int **pkcs1_encrypt** (int block_type, struct crypto_rsa_key *key, int use_private, const u8 *in, size_t inlen, u8 *out, size_t *outlen)
- int **pkcs1_v15_private_key_decrypt** (struct crypto_rsa_key *key, const u8 *in, size_t inlen, u8 *out, size_t *outlen)
- int **pkcs1_decrypt_public_key** (struct crypto_rsa_key *key, const u8 *crypt, size_t crypt_len, u8 *plain, size_t *plain_len)

15.292.1 Detailed Description

PKCS #1 (RSA Encryption).

Copyright

Copyright (c) 2006-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.293 src/tls/pkcs1.h File Reference

PKCS #1 (RSA Encryption).

Functions

- int **pkcs1_encrypt** (int block_type, struct crypto_rsa_key *key, int use_private, const u8 *in, size_t inlen, u8 *out, size_t *outlen)
- int **pkcs1_v15_private_key_decrypt** (struct crypto_rsa_key *key, const u8 *in, size_t inlen, u8 *out, size_t *outlen)
- int **pkcs1_decrypt_public_key** (struct crypto_rsa_key *key, const u8 *crypt, size_t crypt_len, u8 *plain, size_t *plain_len)

15.293.1 Detailed Description

PKCS #1 (RSA Encryption).

Copyright

15.294 src/tls/pkcs5.c File Reference

PKCS #5 (Password-based Encryption). `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

`#include "md5.h"`

`#include "asn1.h"`

`#include "pkcs5.h"`

Data Structures

- struct pkcs5_params

Functions

- enum pkcs5_alg **pkcs5_get_alg** (struct asn1_oid *oid)
- u8 * **pkcs5_decrypt** (const u8 *enc_alg, size_t enc_alg_len, const u8 *enc_data, size_t enc_data_len, const char *passwd, size_t *data_len)

15.294.1 Detailed Description

PKCS #5 (Password-based Encryption).

Copyright

15.295 src/tls/pkcs5.h File Reference

PKCS #5 (Password-based Encryption).

Functions

- u8 * **pkcs5_decrypt** (const u8 *enc_alg, size_t enc_alg_len, const u8 *enc_data, size_t enc_data_len, const char *passwd, size_t *data_len)

15.295.1 Detailed Description

PKCS #5 (Password-based Encryption).

Copyright

15.296 src/tls/pkcs8.c File Reference

PKCS #8 (Private-key information syntax). `#include "includes.h"`

```
#include "common.h"
#include "asn1.h"
#include "bignum.h"
#include "rsa.h"
#include "pkcs5.h"
#include "pkcs8.h"
```

Functions

- struct crypto_private_key * **pkcs8_key_import** (const u8 *buf, size_t len)
- struct crypto_private_key * **pkcs8_enc_key_import** (const u8 *buf, size_t len, const char *passwd)

15.296.1 Detailed Description

PKCS #8 (Private-key information syntax).

Copyright

Copyright (c) 2006-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.297 src/tls/pkcs8.h File Reference

PKCS #8 (Private-key information syntax).

Functions

- struct crypto_private_key * **pkcs8_key_import** (const u8 *buf, size_t len)
- struct crypto_private_key * **pkcs8_enc_key_import** (const u8 *buf, size_t len, const char *passwd)

15.297.1 Detailed Description

PKCS #8 (Private-key information syntax).

Copyright

15.298 src/tls/rsa.c File Reference

RSA. `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

`#include "asn1.h"`

`#include "bignum.h"`

`#include "rsa.h"`

Data Structures

- struct crypto_rsa_key

Functions

- size_t crypto_rsa_get_modulus_len (struct crypto_rsa_key *key)

  Get the modulus length of the RSA key.

- int crypto_rsa_exptmod (const u8 *in, size_t inlen, u8 *out, size_t *outlen, struct crypto_rsa_key *key, int use_private)

  RSA modular exponentiation.

- void crypto_rsa_free (struct crypto_rsa_key *key)

  Free RSA key.

15.298.1 Detailed Description

RSA.

Copyright

Copyright (c) 2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.298.2 Function Documentation

15.298.2.1 int crypto_rsa_exptmod (const u8 * *in*, size_t *inlen*, u8 * *out*, size_t * *outlen*, struct crypto_rsa_key * *key*, int *use_private*)

RSA modular exponentiation.

Parameters:

in Input data

inlen Input data length

out Buffer for output data

outlen Maximum size of the output buffer and used size on success

key RSA key

use_private 1 = Use RSA private key, 0 = Use RSA public key

Returns:

0 on success, -1 on failure

15.298.2.2 void crypto_rsa_free (struct crypto_rsa_key * *key*)

Free RSA key.

Parameters:

key RSA key to be freed

This function frees an RSA key imported with either crypto_rsa_import_public_key() or crypto_rsa_-import_private_key().

15.298.2.3 size_t crypto_rsa_get_modulus_len (struct crypto_rsa_key * *key*)

Get the modulus length of the RSA key.

Parameters:

key RSA key

Returns:

Modulus length of the key

15.299 src/tls/rsa.h File Reference

RSA.

Functions

- struct crypto_rsa_key * **crypto_rsa_import_public_key** (const u8 *buf, size_t len)
- struct crypto_rsa_key * **crypto_rsa_import_private_key** (const u8 *buf, size_t len)
- size_t crypto_rsa_get_modulus_len (struct crypto_rsa_key *key)

  Get the modulus length of the RSA key.

- int crypto_rsa_exptmod (const u8 *in, size_t inlen, u8 *out, size_t *outlen, struct crypto_rsa_key *key, int use_private)

  RSA modular exponentiation.

- void crypto_rsa_free (struct crypto_rsa_key *key)

  Free RSA key.

15.299.1 Detailed Description

RSA.

Copyright

Copyright (c) 2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.299.2 Function Documentation

15.299.2.1 int crypto_rsa_exptmod (const u8 * *in*, size_t *inlen*, u8 * *out*, size_t * *outlen*, struct crypto_rsa_key * *key*, int *use_private*)

RSA modular exponentiation.

Parameters:

in Input data

inlen Input data length

out Buffer for output data

outlen Maximum size of the output buffer and used size on success

key RSA key

use_private 1 = Use RSA private key, 0 = Use RSA public key

Returns:

0 on success, -1 on failure

15.299.2.2 void crypto_rsa_free (struct crypto_rsa_key * *key*)

Free RSA key.

Parameters:

key RSA key to be freed

This function frees an RSA key imported with either crypto_rsa_import_public_key() or crypto_rsa_-import_private_key().

15.299.2.3 size_t crypto_rsa_get_modulus_len (struct crypto_rsa_key * *key*)

Get the modulus length of the RSA key.

Parameters:

key RSA key

Returns:

Modulus length of the key

15.300 src/tls/tlsv1_client.c File Reference

TLSv1 client (RFC 2246). `#include "includes.h"`

```
#include "common.h"
#include "sha1.h"
#include "tls.h"
#include "tlsv1_common.h"
#include "tlsv1_record.h"
#include "tlsv1_client.h"
#include "tlsv1_client_i.h"
```

Functions

- void **tls_alert** (struct tlsv1_client *conn, u8 level, u8 description)
- void **tlsv1_client_free_dh** (struct tlsv1_client *conn)
- int **tls_derive_pre_master_secret** (u8 *pre_master_secret)
- int **tls_derive_keys** (struct tlsv1_client *conn, const u8 *pre_master_secret, size_t pre_master_secret_len)
- u8 * tlsv1_client_handshake (struct tlsv1_client *conn, const u8 *in_data, size_t in_len, size_t *out_len, u8 **appl_data, size_t *appl_data_len)

  Process TLS handshake.

- int tlsv1_client_encrypt (struct tlsv1_client *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Encrypt data into TLS tunnel.

- int tlsv1_client_decrypt (struct tlsv1_client *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Decrypt data from TLS tunnel.

- int tlsv1_client_global_init (void)

  Initialize TLSv1 client.

- void tlsv1_client_global_deinit (void)

  Deinitialize TLSv1 client.

- struct tlsv1_client * tlsv1_client_init (void)

  Initialize TLSv1 client connection.

- void tlsv1_client_deinit (struct tlsv1_client *conn)

  Deinitialize TLSv1 client connection.

- int tlsv1_client_established (struct tlsv1_client *conn)

  Check whether connection has been established.

- int tlsv1_client_prf (struct tlsv1_client *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

Use TLS-PRF to derive keying material.

- int tlsv1_client_get_cipher (struct tlsv1_client *conn, char *buf, size_t buflen)

  Get current cipher name.

- int tlsv1_client_shutdown (struct tlsv1_client *conn)

  Shutdown TLS connection.

- int tlsv1_client_resumed (struct tlsv1_client *conn)

  Was session resumption used.

- int tlsv1_client_hello_ext (struct tlsv1_client *conn, int ext_type, const u8 *data, size_t data_len)

  Set TLS extension for ClientHello.

- int tlsv1_client_get_keys (struct tlsv1_client *conn, struct tls_keys *keys)

  Get master key and random data from TLS connection.

- int tlsv1_client_get_keyblock_size (struct tlsv1_client *conn)

  Get TLS key_block size.

- int tlsv1_client_set_cipher_list (struct tlsv1_client *conn, u8 *ciphers)

  Configure acceptable cipher suites.

- int tlsv1_client_set_cred (struct tlsv1_client *conn, struct tlsv1_credentials *cred)

  Set client credentials.

- void **tlsv1_client_set_session_ticket_cb** (struct tlsv1_client *conn, tlsv1_client_session_ticket_cb cb, void *ctx)

15.300.1 Detailed Description

TLSv1 client (RFC 2246).

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.300.2 Function Documentation

15.300.2.1 int tlsv1_client_decrypt (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

in_data Pointer to input buffer (encrypted TLS data)

in_len Input buffer length

out_data Pointer to output buffer (decrypted data from TLS tunnel)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

15.300.2.2 void tlsv1_client_deinit (struct tlsv1_client * *conn*)

Deinitialize TLSv1 client connection.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

15.300.2.3 int tlsv1_client_encrypt (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

in_data Pointer to plaintext data to be encrypted

in_len Input buffer length

out_data Pointer to output buffer (encrypted TLS data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

15.300.2.4 int tlsv1_client_established (struct tlsv1_client * *conn*)

Check whether connection has been established.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

Returns:

1 if connection is established, 0 if not

15.300.2.5 int tlsv1_client_get_cipher (struct tlsv1_client * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()
buf Buffer for the cipher name
buflen buf size

Returns:

0 on success, -1 on failure

Get the name of the currently used cipher.

15.300.2.6 int tlsv1_client_get_keyblock_size (struct tlsv1_client * *conn*)

Get TLS key_block size.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

Returns:

Size of the key_block for the negotiated cipher suite or -1 on failure

15.300.2.7 int tlsv1_client_get_keys (struct tlsv1_client * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()
keys Structure of key/random data (filled on success)

Returns:

0 on success, -1 on failure

15.300.2.8 void tlsv1_client_global_deinit (void)

Deinitialize TLSv1 client. This function can be used to deinitialize the TLSv1 client that was initialized by calling tlsv1_client_global_init(). No TLSv1 client functions can be called after this before calling tlsv1_client_global_init() again.

15.300.2.9 int tlsv1_client_global_init (void)

Initialize TLSv1 client.

Returns:

0 on success, -1 on failure

This function must be called before using any other TLSv1 client functions.

15.300.2.10 u8* tlsv1_client_handshake (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

appl_data Pointer to application data pointer, or NULL if dropped

appl_data_len Pointer to variable that is set to appl_data length

Returns:

Pointer to output data, NULL on failure

15.300.2.11 int tlsv1_client_hello_ext (struct tlsv1_client * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

ext_type Extension type

data Extension payload (NULL to remove extension)

data_len Extension payload length

Returns:

0 on success, -1 on failure

15.300.2.12 struct tlsv1_client* tlsv1_client_init (void) [read]

Initialize TLSv1 client connection.

Returns:

Pointer to TLSv1 client connection data or NULL on failure

15.300.2.13 int tlsv1_client_prf (struct tlsv1_client * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

15.300.2.14 int tlsv1_client_resumed (struct tlsv1_client * *conn*)

Was session resumption used.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

Returns:

1 if current session used session resumption, 0 if not

15.300.2.15 int tlsv1_client_set_cipher_list (struct tlsv1_client * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.300.2.16 int tlsv1_client_set_cred (struct tlsv1_client * *conn*, struct tlsv1_credentials * *cred*)

Set client credentials.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

cred Credentials from tlsv1_cred_alloc()

Returns:

0 on success, -1 on failure

On success, the client takes ownership of the credentials block and caller must not free it. On failure, caller is responsible for freeing the credential block.

15.300.2.17 int tlsv1_client_shutdown (struct tlsv1_client * *conn*)

Shutdown TLS connection.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

Returns:

0 on success, -1 on failure

15.301 src/tls/tlsv1_client.h File Reference

TLSv1 client (RFC 2246). `#include "tlsv1_cred.h"`

Typedefs

- typedef int(∗ **tlsv1_client_session_ticket_cb**)(void ∗ctx, const u8 ∗ticket, size_t len, const u8 ∗client_random, const u8 ∗server_random, u8 ∗master_secret)

Functions

- int tlsv1_client_global_init (void)

  Initialize TLSv1 client.

- void tlsv1_client_global_deinit (void)

  Deinitialize TLSv1 client.

- struct tlsv1_client ∗ tlsv1_client_init (void)

  Initialize TLSv1 client connection.

- void tlsv1_client_deinit (struct tlsv1_client ∗conn)

  Deinitialize TLSv1 client connection.

- int tlsv1_client_established (struct tlsv1_client ∗conn)

  Check whether connection has been established.

- int tlsv1_client_prf (struct tlsv1_client ∗conn, const char ∗label, int server_random_first, u8 ∗out, size_t out_len)

  Use TLS-PRF to derive keying material.

- u8 ∗ tlsv1_client_handshake (struct tlsv1_client ∗conn, const u8 ∗in_data, size_t in_len, size_t ∗out_len, u8 ∗∗appl_data, size_t ∗appl_data_len)

  Process TLS handshake.

- int tlsv1_client_encrypt (struct tlsv1_client ∗conn, const u8 ∗in_data, size_t in_len, u8 ∗out_data, size_t out_len)

  Encrypt data into TLS tunnel.

- int tlsv1_client_decrypt (struct tlsv1_client ∗conn, const u8 ∗in_data, size_t in_len, u8 ∗out_data, size_t out_len)

  Decrypt data from TLS tunnel.

- int tlsv1_client_get_cipher (struct tlsv1_client ∗conn, char ∗buf, size_t buflen)

  Get current cipher name.

- int tlsv1_client_shutdown (struct tlsv1_client ∗conn)

  Shutdown TLS connection.

- int tlsv1_client_resumed (struct tlsv1_client ∗conn)

Was session resumption used.

- int tlsv1_client_hello_ext (struct tlsv1_client *conn, int ext_type, const u8 *data, size_t data_len)
  Set TLS extension for ClientHello.
- int tlsv1_client_get_keys (struct tlsv1_client *conn, struct tls_keys *keys)
  Get master key and random data from TLS connection.
- int tlsv1_client_get_keyblock_size (struct tlsv1_client *conn)
  Get TLS key_block size.
- int tlsv1_client_set_cipher_list (struct tlsv1_client *conn, u8 *ciphers)
  Configure acceptable cipher suites.
- int tlsv1_client_set_cred (struct tlsv1_client *conn, struct tlsv1_credentials *cred)
  Set client credentials.
- void **tlsv1_client_set_session_ticket_cb** (struct tlsv1_client *conn, tlsv1_client_session_ticket_cb cb, void *ctx)

15.301.1 Detailed Description

TLSv1 client (RFC 2246).

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.301.2 Function Documentation

15.301.2.1 int tlsv1_client_decrypt (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

in_data Pointer to input buffer (encrypted TLS data)

in_len Input buffer length

out_data Pointer to output buffer (decrypted data from TLS tunnel)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to receive data from the encrypted tunnel.

15.301.2.2 void tlsv1_client_deinit (struct tlsv1_client * *conn*)

Deinitialize TLSv1 client connection.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

15.301.2.3 int tlsv1_client_encrypt (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

in_data Pointer to plaintext data to be encrypted

in_len Input buffer length

out_data Pointer to output buffer (encrypted TLS data)

out_len Maximum out_data length

Returns:

Number of bytes written to out_data, -1 on failure

This function is used after TLS handshake has been completed successfully to send data in the encrypted tunnel.

15.301.2.4 int tlsv1_client_established (struct tlsv1_client * *conn*)

Check whether connection has been established.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

Returns:

1 if connection is established, 0 if not

15.301.2.5 int tlsv1_client_get_cipher (struct tlsv1_client * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

buf Buffer for the cipher name

buflen buf size

Returns:

0 on success, -1 on failure

Get the name of the currently used cipher.

15.301.2.6 int tlsv1_client_get_keyblock_size (struct tlsv1_client * *conn*)

Get TLS key_block size.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

Returns:

Size of the key_block for the negotiated cipher suite or -1 on failure

15.301.2.7 int tlsv1_client_get_keys (struct tlsv1_client * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

keys Structure of key/random data (filled on success)

Returns:

0 on success, -1 on failure

15.301.2.8 void tlsv1_client_global_deinit (void)

Deinitialize TLSv1 client. This function can be used to deinitialize the TLSv1 client that was initialized by calling tlsv1_client_global_init(). No TLSv1 client functions can be called after this before calling tlsv1_client_global_init() again.

15.301.2.9 int tlsv1_client_global_init (void)

Initialize TLSv1 client.

Returns:

0 on success, -1 on failure

This function must be called before using any other TLSv1 client functions.

15.301.2.10 u8* tlsv1_client_handshake (struct tlsv1_client * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*, u8 ** *appl_data*, size_t * *appl_data_len*)

Process TLS handshake.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

in_data Input data from TLS peer

in_len Input data length

out_len Length of the output buffer.

appl_data Pointer to application data pointer, or NULL if dropped

appl_data_len Pointer to variable that is set to appl_data length

Returns:

Pointer to output data, NULL on failure

15.301.2.11 int tlsv1_client_hello_ext (struct tlsv1_client * *conn*, int *ext_type*, const u8 * *data*, size_t *data_len*)

Set TLS extension for ClientHello.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

ext_type Extension type

data Extension payload (NULL to remove extension)

data_len Extension payload length

Returns:

0 on success, -1 on failure

15.301.2.12 struct tlsv1_client* tlsv1_client_init (void) [read]

Initialize TLSv1 client connection.

Returns:

Pointer to TLSv1 client connection data or NULL on failure

15.301.2.13 int tlsv1_client_prf (struct tlsv1_client * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

15.301.2.14 int tlsv1_client_resumed (struct tlsv1_client * *conn*)

Was session resumption used.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

Returns:

1 if current session used session resumption, 0 if not

15.301.2.15 int tlsv1_client_set_cipher_list (struct tlsv1_client * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.301.2.16 int tlsv1_client_set_cred (struct tlsv1_client * *conn*, struct tlsv1_credentials * *cred*)

Set client credentials.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

cred Credentials from tlsv1_cred_alloc()

Returns:

0 on success, -1 on failure

On success, the client takes ownership of the credentials block and caller must not free it. On failure, caller is responsible for freeing the credential block.

15.301.2.17 int tlsv1_client_shutdown (struct tlsv1_client * *conn*)

Shutdown TLS connection.

Parameters:

conn TLSv1 client connection data from tlsv1_client_init()

Returns:

0 on success, -1 on failure

15.302 src/tls/tlsv1_client_i.h File Reference

TLSv1 client - internal structures.

Data Structures

- struct tlsv1_client

Defines

- #define **MAX_CIPHER_COUNT** 30

Functions

- void **tls_alert** (struct tlsv1_client *conn, u8 level, u8 description)
- void **tlsv1_client_free_dh** (struct tlsv1_client *conn)
- int **tls_derive_pre_master_secret** (u8 *pre_master_secret)
- int **tls_derive_keys** (struct tlsv1_client *conn, const u8 *pre_master_secret, size_t pre_master_secret_len)
- u8 * **tls_send_client_hello** (struct tlsv1_client *conn, size_t *out_len)
- u8 * **tlsv1_client_send_alert** (struct tlsv1_client *conn, u8 level, u8 description, size_t *out_len)
- u8 * **tlsv1_client_handshake_write** (struct tlsv1_client *conn, size_t *out_len, int no_appl_data)
- int **tlsv1_client_process_handshake** (struct tlsv1_client *conn, u8 ct, const u8 *buf, size_t *len, u8 **out_data, size_t *out_len)

15.302.1 Detailed Description

TLSv1 client - internal structures.

Copyright

15.303 src/tls/tlsv1_client_read.c File Reference

TLSv1 client - read handshake message. `#include "includes.h"`

```
#include "common.h"
#include "md5.h"
#include "sha1.h"
#include "x509v3.h"
#include "tls.h"
#include "tlsv1_common.h"
#include "tlsv1_record.h"
#include "tlsv1_client.h"
#include "tlsv1_client_i.h"
```

Functions

- int **tlsv1_client_process_handshake** (struct tlsv1_client *conn, u8 ct, const u8 *buf, size_t *len, u8 **out_data, size_t *out_len)

15.303.1 Detailed Description

TLSv1 client - read handshake message.

Copyright

15.304 src/tls/tlsv1_client_write.c File Reference

TLSv1 client - write handshake message. `#include "includes.h"`

```
#include "common.h"
#include "md5.h"
#include "sha1.h"
#include "x509v3.h"
#include "tls.h"
#include "tlsv1_common.h"
#include "tlsv1_record.h"
#include "tlsv1_client.h"
#include "tlsv1_client_i.h"
```

Functions

- u8 * **tls_send_client_hello** (struct tlsv1_client *conn, size_t *out_len)
- u8 * **tlsv1_client_handshake_write** (struct tlsv1_client *conn, size_t *out_len, int no_appl_data)
- u8 * **tlsv1_client_send_alert** (struct tlsv1_client *conn, u8 level, u8 description, size_t *out_len)

15.304.1 Detailed Description

TLSv1 client - write handshake message.

Copyright

15.305 src/tls/tlsv1_common.c File Reference

TLSv1 common routines. `#include "includes.h"`

`#include "common.h"`

`#include "x509v3.h"`

`#include "tlsv1_common.h"`

Defines

- #define **NUM_ELEMS**(a) (sizeof(a) / sizeof((a)[0]))
- #define **NUM_TLS_CIPHER_SUITES** NUM_ELEMS(tls_cipher_suites)
- #define **NUM_TLS_CIPHER_DATA** NUM_ELEMS(tls_ciphers)

Functions

- struct tls_cipher_suite * tls_get_cipher_suite (u16 suite)

  Get TLS cipher suite.

- struct tls_cipher_data * **tls_get_cipher_data** (tls_cipher cipher)
- int **tls_server_key_exchange_allowed** (tls_cipher cipher)
- int tls_parse_cert (const u8 *buf, size_t len, struct crypto_public_key **pk)

  Parse DER encoded X.509 certificate and get public key.

- int **tls_verify_hash_init** (struct tls_verify_hash *verify)
- void **tls_verify_hash_add** (struct tls_verify_hash *verify, const u8 *buf, size_t len)
- void **tls_verify_hash_free** (struct tls_verify_hash *verify)

15.305.1 Detailed Description

TLSv1 common routines.

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.305.2 Function Documentation

15.305.2.1 struct tls_cipher_suite* tls_get_cipher_suite (u16 *suite*) [read]

Get TLS cipher suite.

Parameters:

suite Cipher suite identifier

Returns:

Pointer to the cipher data or NULL if not found

15.305.2.2 int tls_parse_cert (const u8 * *buf*, size_t *len*, struct crypto_public_key ** *pk*)

Parse DER encoded X.509 certificate and get public key.

Parameters:

buf ASN.1 DER encoded certificate

len Length of the buffer

pk Buffer for returning the allocated public key

Returns:

0 on success, -1 on failure

This functions parses an ASN.1 DER encoded X.509 certificate and retrieves the public key from it. The caller is responsible for freeing the public key by calling crypto_public_key_free().

15.306 src/tls/tlsv1_common.h File Reference

TLSv1 common definitions. `#include "crypto.h"`

Data Structures

- struct tls_cipher_suite
- struct tls_cipher_data
- struct tls_verify_hash

Defines

- #define **TLS_VERSION** 0x0301
- #define **TLS_RANDOM_LEN** 32
- #define **TLS_PRE_MASTER_SECRET_LEN** 48
- #define **TLS_MASTER_SECRET_LEN** 48
- #define **TLS_SESSION_ID_MAX_LEN** 32
- #define **TLS_VERIFY_DATA_LEN** 12
- #define **TLS_NULL_WITH_NULL_NULL** 0x0000
- #define **TLS_RSA_WITH_NULL_MD5** 0x0001
- #define **TLS_RSA_WITH_NULL_SHA** 0x0002
- #define **TLS_RSA_EXPORT_WITH_RC4_40_MD5** 0x0003
- #define **TLS_RSA_WITH_RC4_128_MD5** 0x0004
- #define **TLS_RSA_WITH_RC4_128_SHA** 0x0005
- #define **TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5** 0x0006
- #define **TLS_RSA_WITH_IDEA_CBC_SHA** 0x0007
- #define **TLS_RSA_EXPORT_WITH_DES40_CBC_SHA** 0x0008
- #define **TLS_RSA_WITH_DES_CBC_SHA** 0x0009
- #define **TLS_RSA_WITH_3DES_EDE_CBC_SHA** 0x000A
- #define **TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA** 0x000B
- #define **TLS_DH_DSS_WITH_DES_CBC_SHA** 0x000C
- #define **TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA** 0x000D
- #define **TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA** 0x000E
- #define **TLS_DH_RSA_WITH_DES_CBC_SHA** 0x000F
- #define **TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA** 0x0010
- #define **TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA** 0x0011
- #define **TLS_DHE_DSS_WITH_DES_CBC_SHA** 0x0012
- #define **TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA** 0x0013
- #define **TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA** 0x0014
- #define **TLS_DHE_RSA_WITH_DES_CBC_SHA** 0x0015
- #define **TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA** 0x0016
- #define **TLS_DH_anon_EXPORT_WITH_RC4_40_MD5** 0x0017
- #define **TLS_DH_anon_WITH_RC4_128_MD5** 0x0018
- #define **TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA** 0x0019
- #define **TLS_DH_anon_WITH_DES_CBC_SHA** 0x001A
- #define **TLS_DH_anon_WITH_3DES_EDE_CBC_SHA** 0x001B
- #define **TLS_RSA_WITH_AES_128_CBC_SHA** 0x002F
- #define **TLS_DH_DSS_WITH_AES_128_CBC_SHA** 0x0030
- #define **TLS_DH_RSA_WITH_AES_128_CBC_SHA** 0x0031

- #define **TLS_DHE_DSS_WITH_AES_128_CBC_SHA** 0x0032
- #define **TLS_DHE_RSA_WITH_AES_128_CBC_SHA** 0x0033
- #define **TLS_DH_anon_WITH_AES_128_CBC_SHA** 0x0034
- #define **TLS_RSA_WITH_AES_256_CBC_SHA** 0x0035
- #define **TLS_DH_DSS_WITH_AES_256_CBC_SHA** 0x0036
- #define **TLS_DH_RSA_WITH_AES_256_CBC_SHA** 0x0037
- #define **TLS_DHE_DSS_WITH_AES_256_CBC_SHA** 0x0038
- #define **TLS_DHE_RSA_WITH_AES_256_CBC_SHA** 0x0039
- #define **TLS_DH_anon_WITH_AES_256_CBC_SHA** 0x003A
- #define **TLS_COMPRESSION_NULL** 0
- #define **TLS_ALERT_LEVEL_WARNING** 1
- #define **TLS_ALERT_LEVEL_FATAL** 2
- #define **TLS_ALERT_CLOSE_NOTIFY** 0
- #define **TLS_ALERT_UNEXPECTED_MESSAGE** 10
- #define **TLS_ALERT_BAD_RECORD_MAC** 20
- #define **TLS_ALERT_DECRYPTION_FAILED** 21
- #define **TLS_ALERT_RECORD_OVERFLOW** 22
- #define **TLS_ALERT_DECOMPRESSION_FAILURE** 30
- #define **TLS_ALERT_HANDSHAKE_FAILURE** 40
- #define **TLS_ALERT_BAD_CERTIFICATE** 42
- #define **TLS_ALERT_UNSUPPORTED_CERTIFICATE** 43
- #define **TLS_ALERT_CERTIFICATE_REVOKED** 44
- #define **TLS_ALERT_CERTIFICATE_EXPIRED** 45
- #define **TLS_ALERT_CERTIFICATE_UNKNOWN** 46
- #define **TLS_ALERT_ILLEGAL_PARAMETER** 47
- #define **TLS_ALERT_UNKNOWN_CA** 48
- #define **TLS_ALERT_ACCESS_DENIED** 49
- #define **TLS_ALERT_DECODE_ERROR** 50
- #define **TLS_ALERT_DECRYPT_ERROR** 51
- #define **TLS_ALERT_EXPORT_RESTRICTION** 60
- #define **TLS_ALERT_PROTOCOL_VERSION** 70
- #define **TLS_ALERT_INSUFFICIENT_SECURITY** 71
- #define **TLS_ALERT_INTERNAL_ERROR** 80
- #define **TLS_ALERT_USER_CANCELED** 90
- #define **TLS_ALERT_NO_RENEGOTIATION** 100
- #define **TLS_ALERT_UNSUPPORTED_EXTENSION** 110
- #define **TLS_ALERT_CERTIFICATE_UNOBTAINABLE** 111
- #define **TLS_ALERT_UNRECOGNIZED_NAME** 112
- #define **TLS_ALERT_BAD_CERTIFICATE_STATUS_RESPONSE** 113
- #define **TLS_ALERT_BAD_CERTIFICATE_HASH_VALUE** 114
- #define **TLS_EXT_SERVER_NAME** 0
- #define **TLS_EXT_MAX_FRAGMENT_LENGTH** 1
- #define **TLS_EXT_CLIENT_CERTIFICATE_URL** 2
- #define **TLS_EXT_TRUSTED_CA_KEYS** 3
- #define **TLS_EXT_TRUNCATED_HMAC** 4
- #define **TLS_EXT_STATUS_REQUEST** 5
- #define **TLS_EXT_SESSION_TICKET** 35
- #define **TLS_EXT_PAC_OPAQUE** TLS_EXT_SESSION_TICKET

Enumerations

- enum {

  TLS_HANDSHAKE_TYPE_HELLO_REQUEST = 0, **TLS_HANDSHAKE_TYPE_CLIENT_HELLO** = 1, **TLS_HANDSHAKE_TYPE_SERVER_HELLO** = 2, **TLS_HANDSHAKE_TYPE_NEW_SESSION_TICKET** = 4,

  TLS_HANDSHAKE_TYPE_CERTIFICATE = 11, **TLS_HANDSHAKE_TYPE_SERVER_KEY_EXCHANGE** = 12, **TLS_HANDSHAKE_TYPE_CERTIFICATE_REQUEST** = 13, **TLS_HANDSHAKE_TYPE_SERVER_HELLO_DONE** = 14,

  TLS_HANDSHAKE_TYPE_CERTIFICATE_VERIFY = 15, **TLS_HANDSHAKE_TYPE_CLIENT_KEY_EXCHANGE** = 16, **TLS_HANDSHAKE_TYPE_FINISHED** = 20, **TLS_HANDSHAKE_TYPE_CERTIFICATE_URL** = 21,

  TLS_HANDSHAKE_TYPE_CERTIFICATE_STATUS = 22 }
- enum { **TLS_CHANGE_CIPHER_SPEC** = 1 }
- enum **tls_key_exchange** {

  TLS_KEY_X_NULL, **TLS_KEY_X_RSA**, **TLS_KEY_X_RSA_EXPORT**, **TLS_KEY_X_DH_DSS_EXPORT**,

  TLS_KEY_X_DH_DSS, **TLS_KEY_X_DH_RSA_EXPORT**, **TLS_KEY_X_DH_RSA**, **TLS_KEY_X_DHE_DSS_EXPORT**,

  TLS_KEY_X_DHE_DSS, **TLS_KEY_X_DHE_RSA_EXPORT**, **TLS_KEY_X_DHE_RSA**, **TLS_KEY_X_DH_anon_EXPORT**,

  TLS_KEY_X_DH_anon }
- enum **tls_cipher** {

  TLS_CIPHER_NULL, **TLS_CIPHER_RC4_40**, **TLS_CIPHER_RC4_128**, **TLS_CIPHER_RC2_CBC_40**,

  TLS_CIPHER_IDEA_CBC, **TLS_CIPHER_DES40_CBC**, **TLS_CIPHER_DES_CBC**, **TLS_CIPHER_3DES_EDE_CBC**,

  TLS_CIPHER_AES_128_CBC, **TLS_CIPHER_AES_256_CBC** }
- enum **tls_hash** { **TLS_HASH_NULL**, **TLS_HASH_MD5**, **TLS_HASH_SHA** }
- enum **tls_cipher_type** { **TLS_CIPHER_STREAM**, **TLS_CIPHER_BLOCK** }

Functions

- struct tls_cipher_suite * tls_get_cipher_suite (u16 suite)

  Get TLS cipher suite.
- struct tls_cipher_data * **tls_get_cipher_data** (tls_cipher cipher)
- int **tls_server_key_exchange_allowed** (tls_cipher cipher)
- int tls_parse_cert (const u8 *buf, size_t len, struct crypto_public_key **pk)

  Parse DER encoded X.509 certificate and get public key.
- int **tls_verify_hash_init** (struct tls_verify_hash *verify)
- void **tls_verify_hash_add** (struct tls_verify_hash *verify, const u8 *buf, size_t len)
- void **tls_verify_hash_free** (struct tls_verify_hash *verify)

15.306.1 Detailed Description

TLSv1 common definitions.

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.306.2 Function Documentation

15.306.2.1 struct tls_cipher_suite* tls_get_cipher_suite (u16 *suite*) [read]

Get TLS cipher suite.

Parameters:

suite Cipher suite identifier

Returns:

Pointer to the cipher data or NULL if not found

15.306.2.2 int tls_parse_cert (const u8 * *buf*, size_t *len*, struct crypto_public_key ** *pk*)

Parse DER encoded X.509 certificate and get public key.

Parameters:

buf ASN.1 DER encoded certificate

len Length of the buffer

pk Buffer for returning the allocated public key

Returns:

0 on success, -1 on failure

This functions parses an ASN.1 DER encoded X.509 certificate and retrieves the public key from it. The caller is responsible for freeing the public key by calling crypto_public_key_free().

15.307 src/tls/tlsv1_cred.c File Reference

TLSv1 credentials. `#include "includes.h"`

`#include "common.h"`

`#include "base64.h"`

`#include "crypto.h"`

`#include "x509v3.h"`

`#include "tlsv1_cred.h"`

Functions

- struct tlsv1_credentials * **tlsv1_cred_alloc** (void)
- void **tlsv1_cred_free** (struct tlsv1_credentials *cred)
- int tlsv1_set_ca_cert (struct tlsv1_credentials *cred, const char *cert, const u8 *cert_blob, size_t cert_blob_len, const char *path)

  Set trusted CA certificate(s).

- int tlsv1_set_cert (struct tlsv1_credentials *cred, const char *cert, const u8 *cert_blob, size_t cert_blob_len)

  Set certificate.

- int tlsv1_set_private_key (struct tlsv1_credentials *cred, const char *private_key, const char *private_key_passwd, const u8 *private_key_blob, size_t private_key_blob_len)

  Set private key.

- int tlsv1_set_dhparams (struct tlsv1_credentials *cred, const char *dh_file, const u8 *dh_blob, size_t dh_blob_len)

  Set Diffie-Hellman parameters.

15.307.1 Detailed Description

TLSv1 credentials.

Copyright

Copyright (c) 2006-2009, Jouni Malinen <j@w1.fi>

15.307.2 Function Documentation

15.307.2.1 int tlsv1_set_ca_cert (struct tlsv1_credentials * *cred*, const char * *cert*, const u8 * *cert_blob*, size_t *cert_blob_len*, const char * *path*)

Set trusted CA certificate(s).

Parameters:

cred TLSv1 credentials from tlsv1_cred_alloc()

cert File or reference name for X.509 certificate in PEM or DER format

cert_blob cert as inlined data or NULL if not used

cert_blob_len ca_cert_blob length

path Path to CA certificates (not yet supported)

Returns:

0 on success, -1 on failure

15.307.2.2 int tlsv1_set_cert (struct tlsv1_credentials * *cred*, const char * *cert*, const u8 * *cert_blob*, size_t *cert_blob_len*)

Set certificate.

Parameters:

cred TLSv1 credentials from tlsv1_cred_alloc()

cert File or reference name for X.509 certificate in PEM or DER format

cert_blob cert as inlined data or NULL if not used

cert_blob_len cert_blob length

Returns:

0 on success, -1 on failure

15.307.2.3 int tlsv1_set_dhparams (struct tlsv1_credentials * *cred*, const char * *dh_file*, const u8 * *dh_blob*, size_t *dh_blob_len*)

Set Diffie-Hellman parameters.

Parameters:

cred TLSv1 credentials from tlsv1_cred_alloc()

dh_file File or reference name for the DH params in PEM or DER format

dh_blob DH params as inlined data or NULL if not used

dh_blob_len dh_blob length

Returns:

0 on success, -1 on failure

15.307.2.4 int tlsv1_set_private_key (struct tlsv1_credentials * *cred*, const char * *private_key*, const char * *private_key_passwd*, const u8 * *private_key_blob*, size_t *private_key_blob_len*)

Set private key.

Parameters:

cred TLSv1 credentials from tlsv1_cred_alloc()

private_key File or reference name for the key in PEM or DER format

private_key_passwd Passphrase for decrypted private key, NULL if no passphrase is used.

private_key_blob private_key as inlined data or NULL if not used

private_key_blob_len private_key_blob length

Returns:

0 on success, -1 on failure

15.308 src/tls/tlsv1_cred.h File Reference

TLSv1 credentials.

Data Structures

- struct tlsv1_credentials

Functions

- struct tlsv1_credentials * **tlsv1_cred_alloc** (void)
- void **tlsv1_cred_free** (struct tlsv1_credentials *cred)
- int tlsv1_set_ca_cert (struct tlsv1_credentials *cred, const char *cert, const u8 *cert_blob, size_t cert_blob_len, const char *path)

  Set trusted CA certificate(s).

- int tlsv1_set_cert (struct tlsv1_credentials *cred, const char *cert, const u8 *cert_blob, size_t cert_blob_len)

  Set certificate.

- int tlsv1_set_private_key (struct tlsv1_credentials *cred, const char *private_key, const char *private_key_passwd, const u8 *private_key_blob, size_t private_key_blob_len)

  Set private key.

- int tlsv1_set_dhparams (struct tlsv1_credentials *cred, const char *dh_file, const u8 *dh_blob, size_t dh_blob_len)

  Set Diffie-Hellman parameters.

15.308.1 Detailed Description

TLSv1 credentials.

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

15.308.2 Function Documentation

15.308.2.1 int tlsv1_set_ca_cert (struct tlsv1_credentials * *cred*, const char * *cert*, const u8 * *cert_blob*, size_t *cert_blob_len*, const char * *path*)

Set trusted CA certificate(s).

Parameters:

cred TLSv1 credentials from tlsv1_cred_alloc()

cert File or reference name for X.509 certificate in PEM or DER format

cert_blob cert as inlined data or NULL if not used

cert_blob_len ca_cert_blob length

path Path to CA certificates (not yet supported)

Returns:

0 on success, -1 on failure

15.308.2.2 int tlsv1_set_cert (struct tlsv1_credentials * *cred*, const char * *cert*, const u8 * *cert_blob*, size_t *cert_blob_len*)

Set certificate.

Parameters:

cred TLSv1 credentials from tlsv1_cred_alloc()

cert File or reference name for X.509 certificate in PEM or DER format

cert_blob cert as inlined data or NULL if not used

cert_blob_len cert_blob length

Returns:

0 on success, -1 on failure

15.308.2.3 int tlsv1_set_dhparams (struct tlsv1_credentials * *cred*, const char * *dh_file*, const u8 * *dh_blob*, size_t *dh_blob_len*)

Set Diffie-Hellman parameters.

Parameters:

cred TLSv1 credentials from tlsv1_cred_alloc()

dh_file File or reference name for the DH params in PEM or DER format

dh_blob DH params as inlined data or NULL if not used

dh_blob_len dh_blob length

Returns:

0 on success, -1 on failure

15.308.2.4 int tlsv1_set_private_key (struct tlsv1_credentials * *cred*, const char * *private_key*, const char * *private_key_passwd*, const u8 * *private_key_blob*, size_t *private_key_blob_len*)

Set private key.

Parameters:

cred TLSv1 credentials from tlsv1_cred_alloc()

private_key File or reference name for the key in PEM or DER format

private_key_passwd Passphrase for decrypted private key, NULL if no passphrase is used.

private_key_blob private_key as inlined data or NULL if not used

private_key_blob_len private_key_blob length

Returns:

0 on success, -1 on failure

15.309 src/tls/tlsv1_record.c File Reference

TLSv1 Record Protocol. `#include "includes.h"`

`#include "common.h"`

`#include "md5.h"`

`#include "sha1.h"`

`#include "tlsv1_common.h"`

`#include "tlsv1_record.h"`

Functions

- int tlsv1_record_set_cipher_suite (struct tlsv1_record_layer *rl, u16 cipher_suite)

  TLS record layer: Set cipher suite.

- int tlsv1_record_change_write_cipher (struct tlsv1_record_layer *rl)

  TLS record layer: Change write cipher.

- int tlsv1_record_change_read_cipher (struct tlsv1_record_layer *rl)

  TLS record layer: Change read cipher.

- int tlsv1_record_send (struct tlsv1_record_layer *rl, u8 content_type, u8 *buf, size_t buf_size, size_t payload_len, size_t *out_len)

  TLS record layer: Send a message.

- int tlsv1_record_receive (struct tlsv1_record_layer *rl, const u8 *in_data, size_t in_len, u8 *out_data, size_t *out_len, u8 *alert)

  TLS record layer: Process a received message.

15.309.1 Detailed Description

TLSv1 Record Protocol.

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.309.2 Function Documentation

15.309.2.1 int tlsv1_record_change_read_cipher (struct tlsv1_record_layer * *rl*)

TLS record layer: Change read cipher.

Parameters:

rl Pointer to TLS record layer data

Returns:

0 on success (cipher changed), -1 on failure

This function changes TLS record layer to use the new cipher suite configured with tlsv1_record_set_cipher_suite() for reading.

15.309.2.2 int tlsv1_record_change_write_cipher (struct tlsv1_record_layer * *rl*)

TLS record layer: Change write cipher.

Parameters:

rl Pointer to TLS record layer data

Returns:

0 on success (cipher changed), -1 on failure

This function changes TLS record layer to use the new cipher suite configured with tlsv1_record_set_cipher_suite() for writing.

15.309.2.3 int tlsv1_record_receive (struct tlsv1_record_layer * *rl*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t * *out_len*, u8 * *alert*)

TLS record layer: Process a received message.

Parameters:

rl Pointer to TLS record layer data

in_data Received data

in_len Length of the received data

out_data Buffer for output data (must be at least as long as in_data)

out_len Set to maximum out_data length by caller; used to return the length of the used data

alert Buffer for returning an alert value on failure

Returns:

0 on success, -1 on failure

This function decrypts the received message, verifies HMAC and TLS record layer header.

15.309.2.4 int tlsv1_record_send (struct tlsv1_record_layer * *rl*, u8 *content_type*, u8 * *buf*, size_t *buf_size*, size_t *payload_len*, size_t * *out_len*)

TLS record layer: Send a message.

Parameters:

rl Pointer to TLS record layer data

content_type Content type (TLS_CONTENT_TYPE_∗)

buf Buffer to send (with TLS_RECORD_HEADER_LEN octets reserved in the beginning for record layer to fill in; payload filled in after this and extra space in the end for HMAC).

buf_size Maximum buf size

payload_len Length of the payload

out_len Buffer for returning the used buf length

Returns:

0 on success, -1 on failure

This function fills in the TLS record layer header, adds HMAC, and encrypts the data using the current write cipher.

15.309.2.5 int tlsv1_record_set_cipher_suite (struct tlsv1_record_layer ∗ *rl*, u16 *cipher_suite*)

TLS record layer: Set cipher suite.

Parameters:

rl Pointer to TLS record layer data

cipher_suite New cipher suite

Returns:

0 on success, -1 on failure

This function is used to prepare TLS record layer for cipher suite change. tlsv1_record_change_write_cipher() and tlsv1_record_change_read_cipher() functions can then be used to change the currently used ciphers.

15.310 src/tls/tlsv1_record.h File Reference

TLSv1 Record Protocol. `#include "crypto.h"`

Data Structures

- struct tlsv1_record_layer

Defines

- #define **TLS_MAX_WRITE_MAC_SECRET_LEN** 20
- #define **TLS_MAX_WRITE_KEY_LEN** 32
- #define **TLS_MAX_IV_LEN** 16
- #define **TLS_MAX_KEY_BLOCK_LEN**
- #define **TLS_SEQ_NUM_LEN** 8
- #define **TLS_RECORD_HEADER_LEN** 5

Enumerations

- enum { **TLS_CONTENT_TYPE_CHANGE_CIPHER_SPEC** = 20, **TLS_CONTENT_TYPE_ALERT** = 21, **TLS_CONTENT_TYPE_HANDSHAKE** = 22, **TLS_CONTENT_TYPE_APPLICATION_DATA** = 23 }

Functions

- int tlsv1_record_set_cipher_suite (struct tlsv1_record_layer *rl, u16 cipher_suite)

  TLS record layer: Set cipher suite.

- int tlsv1_record_change_write_cipher (struct tlsv1_record_layer *rl)

  TLS record layer: Change write cipher.

- int tlsv1_record_change_read_cipher (struct tlsv1_record_layer *rl)

  TLS record layer: Change read cipher.

- int tlsv1_record_send (struct tlsv1_record_layer *rl, u8 content_type, u8 *buf, size_t buf_size, size_t payload_len, size_t *out_len)

  TLS record layer: Send a message.

- int tlsv1_record_receive (struct tlsv1_record_layer *rl, const u8 *in_data, size_t in_len, u8 *out_data, size_t *out_len, u8 *alert)

  TLS record layer: Process a received message.

15.310.1 Detailed Description

TLSv1 Record Protocol.

Copyright

15.310.2 Define Documentation

15.310.2.1 #define TLS_MAX_KEY_BLOCK_LEN

Value:

```
(2 * (TLS_MAX_WRITE_MAC_SECRET_LEN + \
                                    TLS_MAX_WRITE_KEY_LEN + TLS_MAX_IV_LEN))
```

15.310.3 Function Documentation

15.310.3.1 int tlsv1_record_change_read_cipher (struct tlsv1_record_layer * *rl*)

TLS record layer: Change read cipher.

Parameters:

rl Pointer to TLS record layer data

Returns:

0 on success (cipher changed), -1 on failure

This function changes TLS record layer to use the new cipher suite configured with tlsv1_record_set_cipher_suite() for reading.

15.310.3.2 int tlsv1_record_change_write_cipher (struct tlsv1_record_layer * *rl*)

TLS record layer: Change write cipher.

Parameters:

rl Pointer to TLS record layer data

Returns:

0 on success (cipher changed), -1 on failure

This function changes TLS record layer to use the new cipher suite configured with tlsv1_record_set_cipher_suite() for writing.

15.310.3.3 int tlsv1_record_receive (struct tlsv1_record_layer * *rl*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t * *out_len*, u8 * *alert*)

TLS record layer: Process a received message.

Parameters:

rl Pointer to TLS record layer data

in_data Received data

in_len Length of the received data

out_data Buffer for output data (must be at least as long as in_data)

out_len Set to maximum out_data length by caller; used to return the length of the used data

alert Buffer for returning an alert value on failure

Returns:

0 on success, -1 on failure

This function decrypts the received message, verifies HMAC and TLS record layer header.

15.310.3.4 int tlsv1_record_send (struct tlsv1_record_layer * *rl*, u8 *content_type*, u8 * *buf*, size_t *buf_size*, size_t *payload_len*, size_t * *out_len*)

TLS record layer: Send a message.

Parameters:

rl Pointer to TLS record layer data

content_type Content type (TLS_CONTENT_TYPE_*)

buf Buffer to send (with TLS_RECORD_HEADER_LEN octets reserved in the beginning for record layer to fill in; payload filled in after this and extra space in the end for HMAC).

buf_size Maximum buf size

payload_len Length of the payload

out_len Buffer for returning the used buf length

Returns:

0 on success, -1 on failure

This function fills in the TLS record layer header, adds HMAC, and encrypts the data using the current write cipher.

15.310.3.5 int tlsv1_record_set_cipher_suite (struct tlsv1_record_layer * *rl*, u16 *cipher_suite*)

TLS record layer: Set cipher suite.

Parameters:

rl Pointer to TLS record layer data

cipher_suite New cipher suite

Returns:

0 on success, -1 on failure

This function is used to prepare TLS record layer for cipher suite change. tlsv1_record_change_write_cipher() and tlsv1_record_change_read_cipher() functions can then be used to change the currently used ciphers.

15.311 src/tls/tlsv1_server.c File Reference

TLSv1 server (RFC 2246). `#include "includes.h"`

```
#include "common.h"
#include "sha1.h"
#include "tls.h"
#include "tlsv1_common.h"
#include "tlsv1_record.h"
#include "tlsv1_server.h"
#include "tlsv1_server_i.h"
```

Functions

- void **tlsv1_server_alert** (struct tlsv1_server *conn, u8 level, u8 description)
- int **tlsv1_server_derive_keys** (struct tlsv1_server *conn, const u8 *pre_master_secret, size_t pre_master_secret_len)
- u8 * tlsv1_server_handshake (struct tlsv1_server *conn, const u8 *in_data, size_t in_len, size_t *out_len)

  Process TLS handshake.

- int tlsv1_server_encrypt (struct tlsv1_server *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Encrypt data into TLS tunnel.

- int tlsv1_server_decrypt (struct tlsv1_server *conn, const u8 *in_data, size_t in_len, u8 *out_data, size_t out_len)

  Decrypt data from TLS tunnel.

- int tlsv1_server_global_init (void)

  Initialize TLSv1 server.

- void tlsv1_server_global_deinit (void)

  Deinitialize TLSv1 server.

- struct tlsv1_server * tlsv1_server_init (struct tlsv1_credentials *cred)

  Initialize TLSv1 server connection.

- void tlsv1_server_deinit (struct tlsv1_server *conn)

  Deinitialize TLSv1 server connection.

- int tlsv1_server_established (struct tlsv1_server *conn)

  Check whether connection has been established.

- int tlsv1_server_prf (struct tlsv1_server *conn, const char *label, int server_random_first, u8 *out, size_t out_len)

  Use TLS-PRF to derive keying material.

- int tlsv1_server_get_cipher (struct tlsv1_server *conn, char *buf, size_t buflen)
  Get current cipher name.
- int tlsv1_server_shutdown (struct tlsv1_server *conn)
  Shutdown TLS connection.
- int tlsv1_server_resumed (struct tlsv1_server *conn)
  Was session resumption used.
- int tlsv1_server_get_keys (struct tlsv1_server *conn, struct tls_keys *keys)
  Get master key and random data from TLS connection.
- int tlsv1_server_get_keyblock_size (struct tlsv1_server *conn)
  Get TLS key_block size.
- int tlsv1_server_set_cipher_list (struct tlsv1_server *conn, u8 *ciphers)
  Configure acceptable cipher suites.
- int **tlsv1_server_set_verify** (struct tlsv1_server *conn, int verify_peer)
- void **tlsv1_server_set_session_ticket_cb** (struct tlsv1_server *conn, tlsv1_server_session_ticket_cb cb, void *ctx)

15.311.1 Detailed Description

TLSv1 server (RFC 2246).

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.311.2 Function Documentation

15.311.2.1 int tlsv1_server_decrypt (struct tlsv1_server * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

Parameters:

conn TLSv1 server connection data from tlsv1_server_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.

15.311.2.2 void tlsv1_server_deinit (struct tlsv1_server * *conn*)

Deinitialize TLSv1 server connection.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

15.311.2.3 int tlsv1_server_encrypt (struct tlsv1_server * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

Parameters:

conn TLSv1 server connection data from tlsv1_server_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.

15.311.2.4 int tlsv1_server_established (struct tlsv1_server * *conn*)

Check whether connection has been established.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

Returns:

1 if connection is established, 0 if not

15.311.2.5 int tlsv1_server_get_cipher (struct tlsv1_server * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

Parameters:

conn TLSv1 server connection data from tlsv1_server_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.

15.311.2.6 int tlsv1_server_get_keyblock_size (struct tlsv1_server * *conn*)

Get TLS key_block size.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

Returns:

Size of the key_block for the negotiated cipher suite or -1 on failure

15.311.2.7 int tlsv1_server_get_keys (struct tlsv1_server * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

keys Structure of key/random data (filled on success)

Returns:

0 on success, -1 on failure

15.311.2.8 void tlsv1_server_global_deinit (void)

Deinitialize TLSv1 server. This function can be used to deinitialize the TLSv1 server that was initialized by calling tlsv1_server_global_init(). No TLSv1 server functions can be called after this before calling tlsv1_server_global_init() again.

15.311.2.9 int tlsv1_server_global_init (void)

Initialize TLSv1 server.

Returns:

0 on success, -1 on failure

This function must be called before using any other TLSv1 server functions.

15.311.2.10 u8* tlsv1_server_handshake (struct tlsv1_server * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake.

Parameters:

conn TLSv1 server connection data from tlsv1_server_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

15.311.2.11 struct tlsv1_server* tlsv1_server_init (struct tlsv1_credentials * *cred*) [read]

Initialize TLSv1 server connection.

Parameters:

cred Pointer to server credentials from tlsv1_server_cred_alloc()

Returns:

Pointer to TLSv1 server connection data or NULL on failure

15.311.2.12 int tlsv1_server_prf (struct tlsv1_server * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

15.311.2.13 int tlsv1_server_resumed (struct tlsv1_server * *conn*)

Was session resumption used.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

Returns:

1 if current session used session resumption, 0 if not

15.311.2.14 int tlsv1_server_set_cipher_list (struct tlsv1_server * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.311.2.15 int tlsv1_server_shutdown (struct tlsv1_server * *conn*)

Shutdown TLS connection.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

Returns:

0 on success, -1 on failure

15.312 src/tls/tlsv1_server.h File Reference

TLSv1 server (RFC 2246). `#include "tlsv1_cred.h"`

Typedefs

- typedef int(∗ **tlsv1_server_session_ticket_cb**)(void ∗ctx, const u8 ∗ticket, size_t len, const u8 ∗client_random, const u8 ∗server_random, u8 ∗master_secret)

Functions

- int tlsv1_server_global_init (void)

  Initialize TLSv1 server.

- void tlsv1_server_global_deinit (void)

  Deinitialize TLSv1 server.

- struct tlsv1_server ∗ tlsv1_server_init (struct tlsv1_credentials ∗cred)

  Initialize TLSv1 server connection.

- void tlsv1_server_deinit (struct tlsv1_server ∗conn)

  Deinitialize TLSv1 server connection.

- int tlsv1_server_established (struct tlsv1_server ∗conn)

  Check whether connection has been established.

- int tlsv1_server_prf (struct tlsv1_server ∗conn, const char ∗label, int server_random_first, u8 ∗out, size_t out_len)

  Use TLS-PRF to derive keying material.

- u8 ∗ tlsv1_server_handshake (struct tlsv1_server ∗conn, const u8 ∗in_data, size_t in_len, size_t ∗out_len)

  Process TLS handshake.

- int tlsv1_server_encrypt (struct tlsv1_server ∗conn, const u8 ∗in_data, size_t in_len, u8 ∗out_data, size_t out_len)

  Encrypt data into TLS tunnel.

- int tlsv1_server_decrypt (struct tlsv1_server ∗conn, const u8 ∗in_data, size_t in_len, u8 ∗out_data, size_t out_len)

  Decrypt data from TLS tunnel.

- int tlsv1_server_get_cipher (struct tlsv1_server ∗conn, char ∗buf, size_t buflen)

  Get current cipher name.

- int tlsv1_server_shutdown (struct tlsv1_server ∗conn)

  Shutdown TLS connection.

- int tlsv1_server_resumed (struct tlsv1_server ∗conn)

Was session resumption used.

- int tlsv1_server_get_keys (struct tlsv1_server *conn, struct tls_keys *keys)

  Get master key and random data from TLS connection.

- int tlsv1_server_get_keyblock_size (struct tlsv1_server *conn)

  Get TLS key_block size.

- int tlsv1_server_set_cipher_list (struct tlsv1_server *conn, u8 *ciphers)

  Configure acceptable cipher suites.

- int **tlsv1_server_set_verify** (struct tlsv1_server *conn, int verify_peer)
- void **tlsv1_server_set_session_ticket_cb** (struct tlsv1_server *conn, tlsv1_server_session_ticket_cb cb, void *ctx)

15.312.1 Detailed Description

TLSv1 server (RFC 2246).

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.312.2 Function Documentation

15.312.2.1 int tlsv1_server_decrypt (struct tlsv1_server * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Decrypt data from TLS tunnel.

Parameters:

conn TLSv1 server connection data from tlsv1_server_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.

15.312.2.2 void tlsv1_server_deinit (struct tlsv1_server * *conn*)

Deinitialize TLSv1 server connection.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

15.312.2.3 int tlsv1_server_encrypt (struct tlsv1_server * *conn*, const u8 * *in_data*, size_t *in_len*, u8 * *out_data*, size_t *out_len*)

Encrypt data into TLS tunnel.

Parameters:

conn TLSv1 server connection data from tlsv1_server_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.

15.312.2.4 int tlsv1_server_established (struct tlsv1_server * *conn*)

Check whether connection has been established.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

Returns:

1 if connection is established, 0 if not

15.312.2.5 int tlsv1_server_get_cipher (struct tlsv1_server * *conn*, char * *buf*, size_t *buflen*)

Get current cipher name.

Parameters:

conn TLSv1 server connection data from tlsv1_server_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.

15.312.2.6 int tlsv1_server_get_keyblock_size (struct tlsv1_server * *conn*)

Get TLS key_block size.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

Returns:

Size of the key_block for the negotiated cipher suite or -1 on failure

15.312.2.7 int tlsv1_server_get_keys (struct tlsv1_server * *conn*, struct tls_keys * *keys*)

Get master key and random data from TLS connection.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

keys Structure of key/random data (filled on success)

Returns:

0 on success, -1 on failure

15.312.2.8 void tlsv1_server_global_deinit (void)

Deinitialize TLSv1 server. This function can be used to deinitialize the TLSv1 server that was initialized by calling tlsv1_server_global_init(). No TLSv1 server functions can be called after this before calling tlsv1_server_global_init() again.

15.312.2.9 int tlsv1_server_global_init (void)

Initialize TLSv1 server.

Returns:

0 on success, -1 on failure

This function must be called before using any other TLSv1 server functions.

15.312.2.10 u8* tlsv1_server_handshake (struct tlsv1_server * *conn*, const u8 * *in_data*, size_t *in_len*, size_t * *out_len*)

Process TLS handshake.

Parameters:

conn TLSv1 server connection data from tlsv1_server_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

15.312.2.11 struct tlsv1_server* tlsv1_server_init (struct tlsv1_credentials * *cred*) [read]

Initialize TLSv1 server connection.

Parameters:

cred Pointer to server credentials from tlsv1_server_cred_alloc()

Returns:

Pointer to TLSv1 server connection data or NULL on failure

15.312.2.12 int tlsv1_server_prf (struct tlsv1_server * *conn*, const char * *label*, int *server_random_first*, u8 * *out*, size_t *out_len*)

Use TLS-PRF to derive keying material.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

label Label (e.g., description of the key) for PRF

server_random_first seed is 0 = client_random|server_random, 1 = server_random|client_random

out Buffer for output data from TLS-PRF

out_len Length of the output buffer

Returns:

0 on success, -1 on failure

15.312.2.13 int tlsv1_server_resumed (struct tlsv1_server * *conn*)

Was session resumption used.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

Returns:

1 if current session used session resumption, 0 if not

15.312.2.14 int tlsv1_server_set_cipher_list (struct tlsv1_server * *conn*, u8 * *ciphers*)

Configure acceptable cipher suites.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

ciphers Zero (TLS_CIPHER_NONE) terminated list of allowed ciphers (TLS_CIPHER_*).

Returns:

0 on success, -1 on failure

15.312.2.15 int tlsv1_server_shutdown (struct tlsv1_server * *conn*)

Shutdown TLS connection.

Parameters:

conn TLSv1 server connection data from tlsv1_server_init()

Returns:

0 on success, -1 on failure

15.313 src/tls/tlsv1_server_i.h File Reference

TLSv1 server - internal structures.

Data Structures

- struct tlsv1_server

Defines

- #define **MAX_CIPHER_COUNT** 30

Functions

- void **tlsv1_server_alert** (struct tlsv1_server *conn, u8 level, u8 description)
- int **tlsv1_server_derive_keys** (struct tlsv1_server *conn, const u8 *pre_master_secret, size_t pre_-master_secret_len)
- u8 * **tlsv1_server_handshake_write** (struct tlsv1_server *conn, size_t *out_len)
- u8 * **tlsv1_server_send_alert** (struct tlsv1_server *conn, u8 level, u8 description, size_t *out_len)
- int **tlsv1_server_process_handshake** (struct tlsv1_server *conn, u8 ct, const u8 *buf, size_t *len)

15.313.1 Detailed Description

TLSv1 server - internal structures.

Copyright

15.314 src/tls/tlsv1_server_read.c File Reference

TLSv1 server - read handshake message. `#include "includes.h"`

```
#include "common.h"
#include "md5.h"
#include "sha1.h"
#include "x509v3.h"
#include "tls.h"
#include "tlsv1_common.h"
#include "tlsv1_record.h"
#include "tlsv1_server.h"
#include "tlsv1_server_i.h"
```

Functions

- int **tlsv1_server_process_handshake** (struct tlsv1_server *conn, u8 ct, const u8 *buf, size_t *len)

15.314.1 Detailed Description

TLSv1 server - read handshake message.

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

15.315 src/tls/tlsv1_server_write.c File Reference

TLSv1 server - write handshake message. `#include "includes.h"`

```
#include "common.h"
#include "md5.h"
#include "sha1.h"
#include "x509v3.h"
#include "tls.h"
#include "tlsv1_common.h"
#include "tlsv1_record.h"
#include "tlsv1_server.h"
#include "tlsv1_server_i.h"
```

Functions

- u8 * **tlsv1_server_handshake_write** (struct tlsv1_server *conn, size_t *out_len)
- u8 * **tlsv1_server_send_alert** (struct tlsv1_server *conn, u8 level, u8 description, size_t *out_len)

15.315.1 Detailed Description

TLSv1 server - write handshake message.

Copyright

15.316 src/tls/x509v3.c File Reference

X.509v3 certificate parsing and processing (RFC 3280 profile). `#include "includes.h"`

`#include "common.h"`

15.316.1 Detailed Description

X.509v3 certificate parsing and processing (RFC 3280 profile).

Copyright

Copyright (c) 2006-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.317 src/tls/x509v3.h File Reference

X.509v3 certificate parsing and processing. `#include "asn1.h"`

Data Structures

- struct x509_algorithm_identifier
- struct x509_name
- struct x509_certificate

Defines

- #define **X509_EXT_BASIC_CONSTRAINTS** (1 << 0)
- #define **X509_EXT_PATH_LEN_CONSTRAINT** (1 << 1)
- #define **X509_EXT_KEY_USAGE** (1 << 2)
- #define **X509_EXT_SUBJECT_ALT_NAME** (1 << 3)
- #define **X509_EXT_ISSUER_ALT_NAME** (1 << 4)
- #define **X509_KEY_USAGE_DIGITAL_SIGNATURE** (1 << 0)
- #define **X509_KEY_USAGE_NON_REPUDIATION** (1 << 1)
- #define **X509_KEY_USAGE_KEY_ENCIPHERMENT** (1 << 2)
- #define **X509_KEY_USAGE_DATA_ENCIPHERMENT** (1 << 3)
- #define **X509_KEY_USAGE_KEY_AGREEMENT** (1 << 4)
- #define **X509_KEY_USAGE_KEY_CERT_SIGN** (1 << 5)
- #define **X509_KEY_USAGE_CRL_SIGN** (1 << 6)
- #define **X509_KEY_USAGE_ENCIPHER_ONLY** (1 << 7)
- #define **X509_KEY_USAGE_DECIPHER_ONLY** (1 << 8)

Enumerations

- enum {

  X509_VALIDATE_OK, **X509_VALIDATE_BAD_CERTIFICATE**, **X509_VALIDATE_UNSUPPORTED_CERTIFICATE**, **X509_VALIDATE_CERTIFICATE_REVOKED**,

  X509_VALIDATE_CERTIFICATE_EXPIRED, **X509_VALIDATE_CERTIFICATE_UNKNOWN**, **X509_VALIDATE_UNKNOWN_CA** }

15.317.1 Detailed Description

X.509v3 certificate parsing and processing.

Copyright

Copyright (c) 2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.318 src/utils/base64.c File Reference

Base64 encoding/decoding (RFC1341). `#include "includes.h"`

`#include "os.h"`

`#include "base64.h"`

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.

15.318.1 Detailed Description

Base64 encoding/decoding (RFC1341).

Copyright

Copyright (c) 2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.318.2 Function Documentation

15.318.2.1 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.

15.318.2.2 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.

15.319 src/utils/base64.h File Reference

Base64 encoding/decoding (RFC1341).

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.

15.319.1 Detailed Description

Base64 encoding/decoding (RFC1341).

Copyright

Copyright (c) 2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.319.2 Function Documentation

15.319.2.1 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.

15.319.2.2 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.

15.320 src/utils/build_config.h File Reference

wpa_supplicant/hostapd - Build time configuration defines

15.320.1 Detailed Description

wpa_supplicant/hostapd - Build time configuration defines

Copyright

Copyright (c) 2005-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This header file can be used to define configuration defines that were originally defined in Makefile. This is mainly meant for IDE use or for systems that do not have suitable 'make' tool. In these cases, it may be easier to have a single place for defining all the needed C pre-processor defines.

15.321 src/utils/common.c File Reference

wpa_supplicant/hostapd / common helper functions, etc. `#include "includes.h"`

`#include "common.h"`

Functions

- int hwaddr_aton (const char *txt, u8 *addr)
  Convert ASCII string to MAC address.
- int hexstr2bin (const char *hex, u8 *buf, size_t len)
  Convert ASCII hex string into binary data.
- void inc_byte_array (u8 *counter, size_t len)
  Increment arbitrary length byte array by one.
- void **wpa_get_ntp_timestamp** (u8 *buf)
- int wpa_snprintf_hex (char *buf, size_t buf_size, const u8 *data, size_t len)
  Print data as a hex string into a buffer.
- int wpa_snprintf_hex_uppercase (char *buf, size_t buf_size, const u8 *data, size_t len)
  Print data as a upper case hex string into buf.
- const char * wpa_ssid_txt (const u8 *ssid, size_t ssid_len)
  Convert SSID to a printable string.
- void * **__hide_aliasing_typecast** (void *foo)

15.321.1 Detailed Description

wpa_supplicant/hostapd / common helper functions, etc.

Copyright

Copyright (c) 2002-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.321.2 Function Documentation

15.321.2.1 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)

15.321.2.2 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)

15.321.2.3 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.

15.321.2.4 int wpa_snprintf_hex (char * *buf*, size_t *buf_size*, const u8 * *data*, size_t *len*)

Print data as a hex string into a buffer.

Parameters:

buf Memory area to use as the output buffer

buf_size Maximum buffer size in bytes (should be at least 2 * len + 1)

data Data to be printed

len Length of data in bytes

Returns:

Number of bytes written

15.321.2.5 int wpa_snprintf_hex_uppercase (char * *buf*, size_t *buf_size*, const u8 * *data*, size_t *len*)

Print data as a upper case hex string into buf.

Parameters:

buf Memory area to use as the output buffer

buf_size Maximum buffer size in bytes (should be at least 2 * len + 1)

data Data to be printed

len Length of data in bytes

Returns:

Number of bytes written

15.321.2.6 const char* wpa_ssid_txt (const 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.

15.322 src/utils/common.h File Reference

wpa_supplicant/hostapd / common helper functions, etc. `#include "os.h"`

```
#include <stdint.h>
#include "wpa_debug.h"
```

Defines

- #define **WPA_TYPES_DEFINED**
- #define **__LITTLE_ENDIAN** 1234
- #define **__BIG_ENDIAN** 4321
- #define **WPA_BYTE_SWAP_DEFINED**
- #define **WPA_GET_BE16**(a) ((u16) (((a)[0] << 8) | (a)[1]))
- #define **WPA_PUT_BE16**(a, val)
- #define **WPA_GET_LE16**(a) ((u16) (((a)[1] << 8) | (a)[0]))
- #define **WPA_PUT_LE16**(a, val)
- #define **WPA_GET_BE24**(a)
- #define **WPA_PUT_BE24**(a, val)
- #define **WPA_GET_BE32**(a)
- #define **WPA_PUT_BE32**(a, val)
- #define **WPA_GET_LE32**(a)
- #define **WPA_PUT_LE32**(a, val)
- #define **WPA_GET_BE64**(a)
- #define **WPA_PUT_BE64**(a, val)
- #define **WPA_GET_LE64**(a)
- #define **ETH_ALEN** 6
- #define **IFNAMSIZ** 16
- #define **ETH_P_ALL** 0x0003
- #define **ETH_P_PAE** 0x888E
- #define **ETH_P_EAPOL** ETH_P_PAE
- #define **ETH_P_RSN_PREAUTH** 0x88c7
- #define **ETH_P_RRB** 0x890D
- #define **PRINTF_FORMAT**(a, b)
- #define STRUCT_PACKED

  WPA Pairwise Transient Key.

- #define **MAC2STR**(a) (a)[0], (a)[1], (a)[2], (a)[3], (a)[4], (a)[5]
- #define **MACSTR** "%02x:%02x:%02x:%02x:%02x:%02x"
- #define **BIT**(x) (1 << (x))
- #define **__force**
- #define **__bitwise**
- #define **wpa_unicode2ascii_inplace**(s) do { } while (0)
- #define **wpa_strdup_tchar**(s) strdup((s))
- #define **aliasing_hide_typecast**(a, t) (t *) __hide_aliasing_typecast((a))

Typedefs

- typedef uint64_t **u64**
- typedef uint32_t **u32**
- typedef uint16_t **u16**
- typedef uint8_t **u8**
- typedef int64_t **s64**
- typedef int32_t **s32**
- typedef int16_t **s16**
- typedef int8_t **s8**
- typedef u16 __bitwise **be16**
- typedef u16 __bitwise **le16**
- typedef u32 __bitwise **be32**
- typedef u32 __bitwise **le32**
- typedef u64 __bitwise **be64**
- typedef u64 __bitwise **le64**

Functions

- int hwaddr_aton (const char *txt, u8 *addr)
  Convert ASCII string to MAC address.
- int hexstr2bin (const char *hex, u8 *buf, size_t len)
  Convert ASCII hex string into binary data.
- void inc_byte_array (u8 *counter, size_t len)
  Increment arbitrary length byte array by one.
- void **wpa_get_ntp_timestamp** (u8 *buf)
- int wpa_snprintf_hex (char *buf, size_t buf_size, const u8 *data, size_t len)
  Print data as a hex string into a buffer.
- int wpa_snprintf_hex_uppercase (char *buf, size_t buf_size, const u8 *data, size_t len)
  Print data as a upper case hex string into buf.
- const char * wpa_ssid_txt (const u8 *ssid, size_t ssid_len)
  Convert SSID to a printable string.
- void * **__hide_aliasing_typecast** (void *foo)

15.322.1 Detailed Description

wpa_supplicant/hostapd / common helper functions, etc.

Copyright

15.322.2 Define Documentation

15.322.2.1 struct radius_attr_vendor STRUCT_PACKED

WPA Pairwise Transient Key. IEEE Std 802.11i-2004 - 8.5.1.2 Pairwise key hierarchy

15.322.2.2 #define WPA_GET_BE24(a)

Value:

```
((((u32) (a)[0]) << 16) | (((u32) (a)[1]) << 8) | \
                          ((u32) (a)[2]))
```

15.322.2.3 #define WPA_GET_BE32(a)

Value:

```
((((u32) (a)[0]) << 24) | (((u32) (a)[1]) << 16) | \
                          (((u32) (a)[2]) << 8) | ((u32) (a)[3]))
```

15.322.2.4 #define WPA_GET_BE64(a)

Value:

```
((((u64) (a)[0]) << 56) | (((u64) (a)[1]) << 48) | \
                          (((u64) (a)[2]) << 40) | (((u64) (a)[3]) << 32) | \
                          (((u64) (a)[4]) << 24) | (((u64) (a)[5]) << 16) | \
                          (((u64) (a)[6]) << 8) | ((u64) (a)[7]))
```

15.322.2.5 #define WPA_GET_LE32(a)

Value:

```
((((u32) (a)[3]) << 24) | (((u32) (a)[2]) << 16) | \
                          (((u32) (a)[1]) << 8) | ((u32) (a)[0]))
```

15.322.2.6 #define WPA_GET_LE64(a)

Value:

```
((((u64) (a)[7]) << 56) | (((u64) (a)[6]) << 48) | \
                          (((u64) (a)[5]) << 40) | (((u64) (a)[4]) << 32) | \
                          (((u64) (a)[3]) << 24) | (((u64) (a)[2]) << 16) | \
                          (((u64) (a)[1]) << 8) | ((u64) (a)[0]))
```

15.322.2.7 #define WPA_PUT_BE16(a, val)

Value:

```
do {                                    \
                (a)[0] = ((u16) (val)) >> 8;    \
                (a)[1] = ((u16) (val)) & 0xff;  \
        } while (0)
```

15.322.2.8 #define WPA_PUT_BE24(a, val)

Value:

```
do {                                                \
                (a)[0] = (u8) ((((u32) (val)) >> 16) & 0xff);   \
                (a)[1] = (u8) ((((u32) (val)) >> 8) & 0xff);    \
                (a)[2] = (u8) (((u32) (val)) & 0xff);           \
        } while (0)
```

15.322.2.9 #define WPA_PUT_BE32(a, val)

Value:

```
do {                                                \
                (a)[0] = (u8) ((((u32) (val)) >> 24) & 0xff);   \
                (a)[1] = (u8) ((((u32) (val)) >> 16) & 0xff);   \
                (a)[2] = (u8) ((((u32) (val)) >> 8) & 0xff);    \
                (a)[3] = (u8) (((u32) (val)) & 0xff);           \
        } while (0)
```

15.322.2.10 #define WPA_PUT_BE64(a, val)

Value:

```
do {                                            \
                (a)[0] = (u8) (((u64) (val)) >> 56);    \
                (a)[1] = (u8) (((u64) (val)) >> 48);    \
                (a)[2] = (u8) (((u64) (val)) >> 40);    \
                (a)[3] = (u8) (((u64) (val)) >> 32);    \
                (a)[4] = (u8) (((u64) (val)) >> 24);    \
                (a)[5] = (u8) (((u64) (val)) >> 16);    \
                (a)[6] = (u8) (((u64) (val)) >> 8);     \
                (a)[7] = (u8) (((u64) (val)) & 0xff);   \
        } while (0)
```

15.322.2.11 #define WPA_PUT_LE16(a, val)

Value:

```
do {                                    \
                (a)[1] = ((u16) (val)) >> 8;    \
                (a)[0] = ((u16) (val)) & 0xff;  \
        } while (0)
```

15.322.2.12 #define WPA_PUT_LE32(a, val)

Value:

```
do {                                                \
                (a)[3] = (u8) ((((u32) (val)) >> 24) & 0xff);   \
                (a)[2] = (u8) ((((u32) (val)) >> 16) & 0xff);   \
                (a)[1] = (u8) ((((u32) (val)) >> 8) & 0xff);    \
                (a)[0] = (u8) (((u32) (val)) & 0xff);           \
        } while (0)
```

15.322.3 Function Documentation

15.322.3.1 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)

15.322.3.2 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)

15.322.3.3 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.

15.322.3.4 int wpa_snprintf_hex (char * *buf*, size_t *buf_size*, const u8 * *data*, size_t *len*)

Print data as a hex string into a buffer.

Parameters:

buf Memory area to use as the output buffer
buf_size Maximum buffer size in bytes (should be at least 2 * len + 1)
data Data to be printed
len Length of data in bytes

Returns:

Number of bytes written

15.322.3.5 int wpa_snprintf_hex_uppercase (char * *buf*, size_t *buf_size*, const u8 * *data*, size_t *len*)

Print data as a upper case hex string into buf.

Parameters:

buf Memory area to use as the output buffer

buf_size Maximum buffer size in bytes (should be at least 2 * len + 1)

data Data to be printed

len Length of data in bytes

Returns:

Number of bytes written

15.322.3.6 const char* wpa_ssid_txt (const 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.

15.323 src/utils/eloop.c File Reference

Event loop based on select() loop. `#include "includes.h"`

`#include "common.h"`

`#include "eloop.h"`

Data Structures

- struct eloop_sock
- struct eloop_timeout
- struct eloop_signal
- struct eloop_sock_table
- struct eloop_data

Functions

- int eloop_init (void *user_data)

  Initialize global event loop data.

- int eloop_register_read_sock (int sock, eloop_sock_handler handler, void *eloop_data, void *user_data)

  Register handler for read events.

- void eloop_unregister_read_sock (int sock)

  Unregister handler for read events.

- int eloop_register_sock (int sock, eloop_event_type type, eloop_sock_handler handler, void *eloop_data, void *user_data)

  Register handler for socket events.

- void eloop_unregister_sock (int sock, eloop_event_type type)

  Unregister handler for socket events.

- int eloop_register_timeout (unsigned int secs, unsigned int usecs, eloop_timeout_handler handler, void *eloop_data, void *user_data)

  Register timeout.

- int eloop_cancel_timeout (eloop_timeout_handler handler, void *eloop_data, void *user_data)

  Cancel timeouts.

- int eloop_is_timeout_registered (eloop_timeout_handler handler, void *eloop_data, void *user_data)

  Check if a timeout is already registered.

- int eloop_register_signal (int sig, eloop_signal_handler handler, void *user_data)

  Register handler for signals.

- int eloop_register_signal_terminate (eloop_signal_handler handler, void *user_data)

Register handler for terminate signals.

- int eloop_register_signal_reconfig (eloop_signal_handler handler, void *user_data)

  Register handler for reconfig signals.

- void eloop_run (void)

  Start the event loop.

- void eloop_terminate (void)

  Terminate event loop.

- void eloop_destroy (void)

  Free any resources allocated for the event loop.

- int eloop_terminated (void)

  Check whether event loop has been terminated.

- void eloop_wait_for_read_sock (int sock)

  Wait for a single reader.

- void * eloop_get_user_data (void)

  Get global user data.

15.323.1 Detailed Description

Event loop based on select() loop.

Copyright

Copyright (c) 2002-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.323.2 Function Documentation

15.323.2.1 int eloop_cancel_timeout (eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Cancel timeouts.

Parameters:

handler Matching callback function

eloop_data Matching eloop_data or ELOOP_ALL_CTX to match all

user_data Matching user_data or ELOOP_ALL_CTX to match all

Returns:

Number of cancelled timeouts

Cancel matching <handler,eloop_data,user_data> timeouts registered with eloop_register_timeout(). ELOOP_ALL_CTX can be used as a wildcard for cancelling all timeouts regardless of eloop_data/user_data.

15.323.2.2 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().

15.323.2.3 void* eloop_get_user_data (void)

Get global user data.

Returns:

user_data pointer that was registered with eloop_init()

15.323.2.4 int eloop_init (void * *user_data*)

Initialize global event loop data.

Parameters:

user_data Pointer to global data passed as eloop_ctx to signal handlers

Returns:

0 on success, -1 on failure

This function must be called before any other eloop_* function. user_data can be used to configure a global (to the process) pointer that will be passed as eloop_ctx parameter to signal handlers.

15.323.2.5 int eloop_is_timeout_registered (eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Check if a timeout is already registered.

Parameters:

handler Matching callback function

eloop_data Matching eloop_data

user_data Matching user_data

Returns:

1 if the timeout is registered, 0 if the timeout is not registered

Determine if a matching <handler,eloop_data,user_data> timeout is registered with eloop_register_timeout().

15.323.2.6 int eloop_register_read_sock (int *sock*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for read events.

Parameters:

sock File descriptor number for the socket

handler Callback function to be called when data is available for reading

eloop_data Callback context data (eloop_ctx)

user_data Callback context data (sock_ctx)

Returns:

0 on success, -1 on failure

Register a read socket notifier for the given file descriptor. The handler function will be called whenever data is available for reading from the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

15.323.2.7 int eloop_register_signal (int *sig*, eloop_signal_handler *handler*, void * *user_data*)

Register handler for signals.

Parameters:

sig Signal number (e.g., SIGHUP)

handler Callback function to be called when the signal is received

user_data Callback context data (signal_ctx)

Returns:

0 on success, -1 on failure

Register a callback function that will be called when a signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

15.323.2.8 int eloop_register_signal_reconfig (eloop_signal_handler *handler*, void * *user_data*)

Register handler for reconfig signals.

Parameters:

handler Callback function to be called when the signal is received

user_data Callback context data (signal_ctx)

Returns:

0 on success, -1 on failure

Register a callback function that will be called when a reconfiguration / hangup signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers a handler for SIGHUP.

15.323.2.9 int eloop_register_signal_terminate (eloop_signal_handler *handler*, void * *user_data*)

Register handler for terminate signals.

Parameters:

handler Callback function to be called when the signal is received

user_data Callback context data (signal_ctx)

Returns:

0 on success, -1 on failure

Register a callback function that will be called when a process termination signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers handlers for SIGINT and SIGTERM.

15.323.2.10 int eloop_register_sock (int *sock*, eloop_event_type *type*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for socket events.

Parameters:

sock File descriptor number for the socket

type Type of event to wait for

handler Callback function to be called when the event is triggered

eloop_data Callback context data (eloop_ctx)

user_data Callback context data (sock_ctx)

Returns:

0 on success, -1 on failure

Register an event notifier for the given socket's file descriptor. The handler function will be called whenever the that event is triggered for the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

15.323.2.11 int eloop_register_timeout (unsigned int *secs*, unsigned int *usecs*, eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Register timeout.

Parameters:

secs Number of seconds to the timeout

usecs Number of microseconds to the timeout

handler Callback function to be called when timeout occurs

eloop_data Callback context data (eloop_ctx)

user_data Callback context data (sock_ctx)

Returns:

0 on success, -1 on failure

Register a timeout that will cause the handler function to be called after given time.

15.323.2.12 void eloop_run (void)

Start the event loop. Start the event loop and continue running as long as there are any registered event handlers. This function is run after event loop has been initialized with event_init() and one or more events have been registered.

15.323.2.13 void eloop_terminate (void)

Terminate event loop. Terminate event loop even if there are registered events. This can be used to request the program to be terminated cleanly.

15.323.2.14 int eloop_terminated (void)

Check whether event loop has been terminated.

Returns:

1 = event loop terminate, 0 = event loop still running

This function can be used to check whether eloop_terminate() has been called to request termination of the event loop. This is normally used to abort operations that may still be queued to be run when eloop_terminate() was called.

15.323.2.15 void eloop_unregister_read_sock (int *sock*)

Unregister handler for read events.

Parameters:

sock File descriptor number for the socket

Unregister a read socket notifier that was previously registered with eloop_register_read_sock().

15.323.2.16 void eloop_unregister_sock (int *sock*, eloop_event_type *type*)

Unregister handler for socket events.

Parameters:

sock File descriptor number for the socket

type Type of event for which sock was registered

Unregister a socket event notifier that was previously registered with eloop_register_sock().

15.323.2.17 void eloop_wait_for_read_sock (int *sock*)

Wait for a single reader.

Parameters:

sock File descriptor number for the socket

Do a blocking wait for a single read socket.

15.324 src/utils/eloop.h File Reference

Event loop.

Defines

- #define ELOOP_ALL_CTX (void *) -1

  eloop_cancel_timeout() magic number to match all timeouts

Typedefs

- typedef void(* eloop_sock_handler)(int sock, void *eloop_ctx, void *sock_ctx)

  eloop socket event callback type

- typedef void(* eloop_event_handler)(void *eloop_data, void *user_ctx)

  eloop generic event callback type

- typedef void(* eloop_timeout_handler)(void *eloop_data, void *user_ctx)

  eloop timeout event callback type

- typedef void(* eloop_signal_handler)(int sig, void *eloop_ctx, void *signal_ctx)

  eloop signal event callback type

Enumerations

- enum eloop_event_type { **EVENT_TYPE_READ** = 0, **EVENT_TYPE_WRITE**, **EVENT_TYPE_EXCEPTION** }

  eloop socket event type for eloop_register_sock()

Functions

- int eloop_init (void *user_data)

  Initialize global event loop data.

- int eloop_register_read_sock (int sock, eloop_sock_handler handler, void *eloop_data, void *user_data)

  Register handler for read events.

- void eloop_unregister_read_sock (int sock)

  Unregister handler for read events.

- int eloop_register_sock (int sock, eloop_event_type type, eloop_sock_handler handler, void *eloop_data, void *user_data)

  Register handler for socket events.

- void eloop_unregister_sock (int sock, eloop_event_type type)

  Unregister handler for socket events.

- int eloop_register_event (void *event, size_t event_size, eloop_event_handler handler, void *eloop_data, void *user_data)

  Register handler for generic events.

- void eloop_unregister_event (void *event, size_t event_size)

  Unregister handler for a generic event.

- int eloop_register_timeout (unsigned int secs, unsigned int usecs, eloop_timeout_handler handler, void *eloop_data, void *user_data)

  Register timeout.

- int eloop_cancel_timeout (eloop_timeout_handler handler, void *eloop_data, void *user_data)

  Cancel timeouts.

- int eloop_is_timeout_registered (eloop_timeout_handler handler, void *eloop_data, void *user_data)

  Check if a timeout is already registered.

- int eloop_register_signal (int sig, eloop_signal_handler handler, void *user_data)

  Register handler for signals.

- int eloop_register_signal_terminate (eloop_signal_handler handler, void *user_data)

  Register handler for terminate signals.

- int eloop_register_signal_reconfig (eloop_signal_handler handler, void *user_data)

  Register handler for reconfig signals.

- void eloop_run (void)

  Start the event loop.

- void eloop_terminate (void)

  Terminate event loop.

- void eloop_destroy (void)

  Free any resources allocated for the event loop.

- int eloop_terminated (void)

  Check whether event loop has been terminated.

- void eloop_wait_for_read_sock (int sock)

  Wait for a single reader.

- void * eloop_get_user_data (void)

  Get global user data.

15.324.1 Detailed Description

Event loop.

Copyright

Copyright (c) 2002-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file defines an event loop interface that supports processing events from registered timeouts (i.e., do something after N seconds), sockets (e.g., a new packet available for reading), and signals. eloop.c is an implementation of this interface using select() and sockets. This is suitable for most UNIX/POSIX systems. When porting to other operating systems, it may be necessary to replace that implementation with OS specific mechanisms.

15.324.2 Typedef Documentation

15.324.2.1 typedef void(* eloop_event_handler)(void *eloop_data, void *user_ctx)

eloop generic event callback type

Parameters:

eloop_ctx Registered callback context data (eloop_data)

sock_ctx Registered callback context data (user_data)

15.324.2.2 typedef void(* eloop_signal_handler)(int sig, void *eloop_ctx, void *signal_ctx)

eloop signal event callback type

Parameters:

sig Signal number

eloop_ctx Registered callback context data (global user_data from eloop_init() call)

signal_ctx Registered callback context data (user_data from eloop_register_signal(), eloop_register_signal_terminate(), or eloop_register_signal_reconfig() call)

15.324.2.3 typedef void(* eloop_sock_handler)(int sock, void *eloop_ctx, void *sock_ctx)

eloop socket event callback type

Parameters:

sock File descriptor number for the socket

eloop_ctx Registered callback context data (eloop_data)

sock_ctx Registered callback context data (user_data)

15.324.2.4 typedef void(* eloop_timeout_handler)(void *eloop_data, void *user_ctx)

eloop timeout event callback type

Parameters:

eloop_ctx Registered callback context data (eloop_data)
sock_ctx Registered callback context data (user_data)

15.324.3 Enumeration Type Documentation

15.324.3.1 enum eloop_event_type

eloop socket event type for eloop_register_sock()

Parameters:

EVENT_TYPE_READ Socket has data available for reading
EVENT_TYPE_WRITE Socket has room for new data to be written
EVENT_TYPE_EXCEPTION An exception has been reported

15.324.4 Function Documentation

15.324.4.1 int eloop_cancel_timeout (eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Cancel timeouts.

Parameters:

handler Matching callback function
eloop_data Matching eloop_data or ELOOP_ALL_CTX to match all
user_data Matching user_data or ELOOP_ALL_CTX to match all

Returns:

Number of cancelled timeouts

Cancel matching <handler,eloop_data,user_data> timeouts registered with eloop_register_timeout(). ELOOP_ALL_CTX can be used as a wildcard for cancelling all timeouts regardless of eloop_data/user_data.

15.324.4.2 void eloop_destroy (void)

Free any resources allocated for the event loop. After calling eloop_destroy(), other eloop_* functions must not be called before re-running eloop_init().

15.324.4.3 void* eloop_get_user_data (void)

Get global user data.

Returns:

user_data pointer that was registered with eloop_init()

15.324.4.4 int eloop_init (void * *user_data*)

Initialize global event loop data.

Parameters:

user_data Pointer to global data passed as eloop_ctx to signal handlers

Returns:

0 on success, -1 on failure

This function must be called before any other eloop_* function. user_data can be used to configure a global (to the process) pointer that will be passed as eloop_ctx parameter to signal handlers.

15.324.4.5 int eloop_is_timeout_registered (eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Check if a timeout is already registered.

Parameters:

handler Matching callback function

eloop_data Matching eloop_data

user_data Matching user_data

Returns:

1 if the timeout is registered, 0 if the timeout is not registered

Determine if a matching <handler,eloop_data,user_data> timeout is registered with eloop_register_timeout().

15.324.4.6 int eloop_register_event (void * *event*, size_t *event_size*, eloop_event_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for generic events.

Parameters:

event Event to wait (eloop implementation specific)

event_size Size of event data

handler Callback function to be called when event is triggered

eloop_data Callback context data (eloop_data)

user_data Callback context data (user_data)

Returns:

0 on success, -1 on failure

Register an event handler for the given event. This function is used to register eloop implementation specific events which are mainly targetted for operating system specific code (driver interface and l2_packet) since

the portable code will not be able to use such an OS-specific call. The handler function will be called whenever the event is triggered. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

In case of Windows implementation (eloop_win.c), event pointer is of HANDLE type, i.e., void*. The callers are likely to have 'HANDLE h' type variable, and they would call this function with eloop_register_event(h, sizeof(h), ...).

15.324.4.7 int eloop_register_read_sock (int *sock*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for read events.

Parameters:

sock File descriptor number for the socket

handler Callback function to be called when data is available for reading

eloop_data Callback context data (eloop_ctx)

user_data Callback context data (sock_ctx)

Returns:

0 on success, -1 on failure

Register a read socket notifier for the given file descriptor. The handler function will be called whenever data is available for reading from the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

15.324.4.8 int eloop_register_signal (int *sig*, eloop_signal_handler *handler*, void * *user_data*)

Register handler for signals.

Parameters:

sig Signal number (e.g., SIGHUP)

handler Callback function to be called when the signal is received

user_data Callback context data (signal_ctx)

Returns:

0 on success, -1 on failure

Register a callback function that will be called when a signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

15.324.4.9 int eloop_register_signal_reconfig (eloop_signal_handler *handler*, void * *user_data*)

Register handler for reconfig signals.

Parameters:

handler Callback function to be called when the signal is received

user_data Callback context data (signal_ctx)

Returns:

0 on success, -1 on failure

Register a callback function that will be called when a reconfiguration / hangup signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers a handler for SIGHUP.

15.324.4.10 int eloop_register_signal_terminate (eloop_signal_handler *handler*, void * *user_data*)

Register handler for terminate signals.

Parameters:

handler Callback function to be called when the signal is received

user_data Callback context data (signal_ctx)

Returns:

0 on success, -1 on failure

Register a callback function that will be called when a process termination signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers handlers for SIGINT and SIGTERM.

15.324.4.11 int eloop_register_sock (int *sock*, eloop_event_type *type*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for socket events.

Parameters:

sock File descriptor number for the socket

type Type of event to wait for

handler Callback function to be called when the event is triggered

eloop_data Callback context data (eloop_ctx)

user_data Callback context data (sock_ctx)

Returns:

0 on success, -1 on failure

Register an event notifier for the given socket's file descriptor. The handler function will be called whenever the that event is triggered for the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

15.324.4.12 int eloop_register_timeout (unsigned int *secs*, unsigned int *usecs*, eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Register timeout.

Parameters:

secs Number of seconds to the timeout

usecs Number of microseconds to the timeout

handler Callback function to be called when timeout occurs

eloop_data Callback context data (eloop_ctx)

user_data Callback context data (sock_ctx)

Returns:

0 on success, -1 on failure

Register a timeout that will cause the handler function to be called after given time.

15.324.4.13 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.

15.324.4.14 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.

15.324.4.15 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.

15.324.4.16 void eloop_unregister_event (void * *event*, size_t *event_size*)

Unregister handler for a generic event.

Parameters:

event Event to cancel (eloop implementation specific)

event_size Size of event data

Unregister a generic event notifier that was previously registered with eloop_register_event().

15.324.4.17 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().

15.324.4.18 void eloop_unregister_sock (int *sock*, eloop_event_type *type*)

Unregister handler for socket events.

Parameters:

sock File descriptor number for the socket

type Type of event for which sock was registered

Unregister a socket event notifier that was previously registered with eloop_register_sock().

15.324.4.19 void eloop_wait_for_read_sock (int *sock*)

Wait for a single reader.

Parameters:

sock File descriptor number for the socket

Do a blocking wait for a single read socket.

15.325 src/utils/eloop_none.c File Reference

Event loop - empty template (basic structure, but no OS specific operations). `#include "includes.h"`

```
#include "common.h"
```

```
#include "eloop.h"
```

Data Structures

- struct eloop_sock
- struct eloop_timeout
- struct eloop_signal
- struct eloop_data

Functions

- int eloop_init (void *user_data)

  Initialize global event loop data.

- int **eloop_register_read_sock** (int sock, void(*handler)(int sock, void *eloop_ctx, void *sock_ctx), void *eloop_data, void *user_data)
- void eloop_unregister_read_sock (int sock)

  Unregister handler for read events.

- int **eloop_register_timeout** (unsigned int secs, unsigned int usecs, void(*handler)(void *eloop_ctx, void *timeout_ctx), void *eloop_data, void *user_data)
- int **eloop_cancel_timeout** (void(*handler)(void *eloop_ctx, void *sock_ctx), void *eloop_data, void *user_data)
- int **eloop_is_timeout_registered** (void(*handler)(void *eloop_ctx, void *timeout_ctx), void *eloop_data, void *user_data)
- int **eloop_register_signal** (int sig, void(*handler)(int sig, void *eloop_ctx, void *signal_ctx), void *user_data)
- int **eloop_register_signal_terminate** (void(*handler)(int sig, void *eloop_ctx, void *signal_ctx), void *user_data)
- int **eloop_register_signal_reconfig** (void(*handler)(int sig, void *eloop_ctx, void *signal_ctx), void *user_data)
- void eloop_run (void)

  Start the event loop.

- void eloop_terminate (void)

  Terminate event loop.

- void eloop_destroy (void)

  Free any resources allocated for the event loop.

- int eloop_terminated (void)

  Check whether event loop has been terminated.

- void eloop_wait_for_read_sock (int sock)

Wait for a single reader.

- void * eloop_get_user_data (void)

  Get global user data.

15.325.1 Detailed Description

Event loop - empty template (basic structure, but no OS specific operations).

Copyright

Copyright (c) 2002-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.325.2 Function Documentation

15.325.2.1 void eloop_destroy (void)

Free any resources allocated for the event loop. After calling eloop_destroy(), other eloop_* functions must not be called before re-running eloop_init().

15.325.2.2 void* eloop_get_user_data (void)

Get global user data.

Returns:

user_data pointer that was registered with eloop_init()

15.325.2.3 int eloop_init (void * *user_data*)

Initialize global event loop data.

Parameters:

user_data Pointer to global data passed as eloop_ctx to signal handlers

Returns:

0 on success, -1 on failure

This function must be called before any other eloop_* function. user_data can be used to configure a global (to the process) pointer that will be passed as eloop_ctx parameter to signal handlers.

15.325.2.4 void eloop_run (void)

Start the event loop. Start the event loop and continue running as long as there are any registered event handlers. This function is run after event loop has been initialized with event_init() and one or more events have been registered.

15.325.2.5 void eloop_terminate (void)

Terminate event loop. Terminate event loop even if there are registered events. This can be used to request the program to be terminated cleanly.

15.325.2.6 int eloop_terminated (void)

Check whether event loop has been terminated.

Returns:

1 = event loop terminate, 0 = event loop still running

This function can be used to check whether eloop_terminate() has been called to request termination of the event loop. This is normally used to abort operations that may still be queued to be run when eloop_terminate() was called.

15.325.2.7 void eloop_unregister_read_sock (int *sock*)

Unregister handler for read events.

Parameters:

sock File descriptor number for the socket

Unregister a read socket notifier that was previously registered with eloop_register_read_sock().

15.325.2.8 void eloop_wait_for_read_sock (int *sock*)

Wait for a single reader.

Parameters:

sock File descriptor number for the socket

Do a blocking wait for a single read socket.

15.326 src/utils/eloop_win.c File Reference

Event loop based on Windows events and WaitForMultipleObjects. `#include "includes.h"`

`#include <winsock2.h>`

`#include "common.h"`

`#include "eloop.h"`

Data Structures

- struct eloop_sock
- struct eloop_event
- struct eloop_timeout
- struct eloop_signal
- struct eloop_data

Functions

- int eloop_init (void *user_data)

  Initialize global event loop data.

- int eloop_register_read_sock (int sock, eloop_sock_handler handler, void *eloop_data, void *user_data)

  Register handler for read events.

- void eloop_unregister_read_sock (int sock)

  Unregister handler for read events.

- int eloop_register_event (void *event, size_t event_size, eloop_event_handler handler, void *eloop_data, void *user_data)

  Register handler for generic events.

- void eloop_unregister_event (void *event, size_t event_size)

  Unregister handler for a generic event.

- int eloop_register_timeout (unsigned int secs, unsigned int usecs, eloop_timeout_handler handler, void *eloop_data, void *user_data)

  Register timeout.

- int eloop_cancel_timeout (eloop_timeout_handler handler, void *eloop_data, void *user_data)

  Cancel timeouts.

- int eloop_is_timeout_registered (eloop_timeout_handler handler, void *eloop_data, void *user_data)

  Check if a timeout is already registered.

- int eloop_register_signal (int sig, eloop_signal_handler handler, void *user_data)

  Register handler for signals.

- int eloop_register_signal_terminate (eloop_signal_handler handler, void *user_data)
  Register handler for terminate signals.
- int eloop_register_signal_reconfig (eloop_signal_handler handler, void *user_data)
  Register handler for reconfig signals.
- void eloop_run (void)
  Start the event loop.
- void eloop_terminate (void)
  Terminate event loop.
- void eloop_destroy (void)
  Free any resources allocated for the event loop.
- int eloop_terminated (void)
  Check whether event loop has been terminated.
- void eloop_wait_for_read_sock (int sock)
  Wait for a single reader.
- void * eloop_get_user_data (void)
  Get global user data.

15.326.1 Detailed Description

Event loop based on Windows events and WaitForMultipleObjects.

Copyright

Copyright (c) 2002-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.326.2 Function Documentation

15.326.2.1 int eloop_cancel_timeout (eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Cancel timeouts.

Parameters:

handler Matching callback function

eloop_data Matching eloop_data or ELOOP_ALL_CTX to match all

user_data Matching user_data or ELOOP_ALL_CTX to match all

Returns:

Number of cancelled timeouts

Cancel matching <handler,eloop_data,user_data> timeouts registered with eloop_register_timeout(). ELOOP_ALL_CTX can be used as a wildcard for cancelling all timeouts regardless of eloop_data/user_data.

15.326.2.2 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().

15.326.2.3 void* eloop_get_user_data (void)

Get global user data.

Returns:

user_data pointer that was registered with eloop_init()

15.326.2.4 int eloop_init (void * *user_data*)

Initialize global event loop data.

Parameters:

user_data Pointer to global data passed as eloop_ctx to signal handlers

Returns:

0 on success, -1 on failure

This function must be called before any other eloop_* function. user_data can be used to configure a global (to the process) pointer that will be passed as eloop_ctx parameter to signal handlers.

15.326.2.5 int eloop_is_timeout_registered (eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Check if a timeout is already registered.

Parameters:

handler Matching callback function

eloop_data Matching eloop_data

user_data Matching user_data

Returns:

1 if the timeout is registered, 0 if the timeout is not registered

Determine if a matching <handler,eloop_data,user_data> timeout is registered with eloop_register_timeout().

15.326.2.6 int eloop_register_event (void * *event*, size_t *event_size*, eloop_event_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for generic events.

Parameters:

event Event to wait (eloop implementation specific)

event_size Size of event data

handler Callback function to be called when event is triggered

eloop_data Callback context data (eloop_data)

user_data Callback context data (user_data)

Returns:

0 on success, -1 on failure

Register an event handler for the given event. This function is used to register eloop implementation specific events which are mainly targetted for operating system specific code (driver interface and l2_packet) since the portable code will not be able to use such an OS-specific call. The handler function will be called whenever the event is triggered. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

In case of Windows implementation (eloop_win.c), event pointer is of HANDLE type, i.e., void*. The callers are likely to have 'HANDLE h' type variable, and they would call this function with eloop_register_-event(h, sizeof(h), ...).

15.326.2.7 int eloop_register_read_sock (int *sock*, eloop_sock_handler *handler*, void * *eloop_data*, void * *user_data*)

Register handler for read events.

Parameters:

sock File descriptor number for the socket

handler Callback function to be called when data is available for reading

eloop_data Callback context data (eloop_ctx)

user_data Callback context data (sock_ctx)

Returns:

0 on success, -1 on failure

Register a read socket notifier for the given file descriptor. The handler function will be called whenever data is available for reading from the socket. The handler function is responsible for clearing the event after having processed it in order to avoid eloop from calling the handler again for the same event.

15.326.2.8 int eloop_register_signal (int *sig*, eloop_signal_handler *handler*, void * *user_data*)

Register handler for signals.

Parameters:

sig Signal number (e.g., SIGHUP)

handler Callback function to be called when the signal is received

user_data Callback context data (signal_ctx)

Returns:

0 on success, -1 on failure

Register a callback function that will be called when a signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

15.326.2.9 int eloop_register_signal_reconfig (eloop_signal_handler *handler*, void * *user_data*)

Register handler for reconfig signals.

Parameters:

handler Callback function to be called when the signal is received

user_data Callback context data (signal_ctx)

Returns:

0 on success, -1 on failure

Register a callback function that will be called when a reconfiguration / hangup signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers a handler for SIGHUP.

15.326.2.10 int eloop_register_signal_terminate (eloop_signal_handler *handler*, void * *user_data*)

Register handler for terminate signals.

Parameters:

handler Callback function to be called when the signal is received

user_data Callback context data (signal_ctx)

Returns:

0 on success, -1 on failure

Register a callback function that will be called when a process termination signal is received. The callback function is actually called only after the system signal handler has returned. This means that the normal limits for sighandlers (i.e., only "safe functions" allowed) do not apply for the registered callback.

Signals are 'global' events and there is no local eloop_data pointer like with other handlers. The global user_data pointer registered with eloop_init() will be used as eloop_ctx for signal handlers.

This function is a more portable version of eloop_register_signal() since the knowledge of exact details of the signals is hidden in eloop implementation. In case of operating systems using signal(), this function registers handlers for SIGINT and SIGTERM.

15.326.2.11 int eloop_register_timeout (unsigned int *secs*, unsigned int *usecs*, eloop_timeout_handler *handler*, void * *eloop_data*, void * *user_data*)

Register timeout.

Parameters:

secs Number of seconds to the timeout

usecs Number of microseconds to the timeout

handler Callback function to be called when timeout occurs

eloop_data Callback context data (eloop_ctx)

user_data Callback context data (sock_ctx)

Returns:

0 on success, -1 on failure

Register a timeout that will cause the handler function to be called after given time.

15.326.2.12 void eloop_run (void)

Start the event loop. Start the event loop and continue running as long as there are any registered event handlers. This function is run after event loop has been initialized with event_init() and one or more events have been registered.

15.326.2.13 void eloop_terminate (void)

Terminate event loop. Terminate event loop even if there are registered events. This can be used to request the program to be terminated cleanly.

15.326.2.14 int eloop_terminated (void)

Check whether event loop has been terminated.

Returns:

1 = event loop terminate, 0 = event loop still running

This function can be used to check whether eloop_terminate() has been called to request termination of the event loop. This is normally used to abort operations that may still be queued to be run when eloop_terminate() was called.

15.326.2.15 void eloop_unregister_event (void * *event*, size_t *event_size*)

Unregister handler for a generic event.

Parameters:

event Event to cancel (eloop implementation specific)

event_size Size of event data

Unregister a generic event notifier that was previously registered with eloop_register_event().

15.326.2.16 void eloop_unregister_read_sock (int *sock*)

Unregister handler for read events.

Parameters:

sock File descriptor number for the socket

Unregister a read socket notifier that was previously registered with eloop_register_read_sock().

15.326.2.17 void eloop_wait_for_read_sock (int *sock*)

Wait for a single reader.

Parameters:

sock File descriptor number for the socket

Do a blocking wait for a single read socket.

15.327 src/utils/includes.h File Reference

wpa_supplicant/hostapd - Default include files `#include "build_config.h"`

```
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include <signal.h>
#include <sys/types.h>
#include <errno.h>
#include <ctype.h>
#include <time.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/uio.h>
#include <sys/time.h>
```

15.327.1 Detailed Description

wpa_supplicant/hostapd - Default include files

Copyright

This header file is included into all C files so that commonly used header files can be selected with OS specific ifdef blocks in one place instead of having to have OS/C library specific selection in many files.

15.328 src/utils/ip_addr.c File Reference

IP address processing. `#include "includes.h"`

`#include "common.h"`

`#include "ip_addr.h"`

Functions

- const char * **hostapd_ip_txt** (const struct hostapd_ip_addr *addr, char *buf, size_t buflen)
- int **hostapd_ip_diff** (struct hostapd_ip_addr *a, struct hostapd_ip_addr *b)
- int **hostapd_parse_ip_addr** (const char *txt, struct hostapd_ip_addr *addr)

15.328.1 Detailed Description

IP address processing.

Copyright

Copyright (c) 2003-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.329 src/utils/ip_addr.h File Reference

IP address processing.

Data Structures

- struct hostapd_ip_addr

Functions

- const char * **hostapd_ip_txt** (const struct hostapd_ip_addr *addr, char *buf, size_t buflen)
- int **hostapd_ip_diff** (struct hostapd_ip_addr *a, struct hostapd_ip_addr *b)
- int **hostapd_parse_ip_addr** (const char *txt, struct hostapd_ip_addr *addr)

15.329.1 Detailed Description

IP address processing.

Copyright

15.330 src/utils/os.h File Reference

wpa_supplicant/hostapd / OS specific functions

Data Structures

- struct os_time

Defines

- #define **os_time_before**(a, b)
- #define **os_time_sub**(a, b, res)
- #define **os_malloc**(s) malloc((s))
- #define **os_realloc**(p, s) realloc((p), (s))
- #define **os_free**(p) free((p))
- #define **os_memcpy**(d, s, n) memcpy((d), (s), (n))
- #define **os_memmove**(d, s, n) memmove((d), (s), (n))
- #define **os_memset**(s, c, n) memset(s, c, n)
- #define **os_memcmp**(s1, s2, n) memcmp((s1), (s2), (n))
- #define **os_strdup**(s) strdup(s)
- #define **os_strlen**(s) strlen(s)
- #define **os_strcasecmp**(s1, s2) strcasecmp((s1), (s2))
- #define **os_strncasecmp**(s1, s2, n) strncasecmp((s1), (s2), (n))
- #define **os_strchr**(s, c) strchr((s), (c))
- #define **os_strcmp**(s1, s2) strcmp((s1), (s2))
- #define **os_strncmp**(s1, s2, n) strncmp((s1), (s2), (n))
- #define **os_strncpy**(d, s, n) strncpy((d), (s), (n))
- #define **os_strrchr**(s, c) strrchr((s), (c))
- #define **os_strstr**(h, n) strstr((h), (n))
- #define **os_snprintf** snprintf

Typedefs

- typedef long **os_time_t**

Functions

- void os_sleep (os_time_t sec, os_time_t usec)

  Sleep (sec, usec).

- int os_get_time (struct os_time *t)

  Get current time (sec, usec).

- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)

  Convert broken-down time into seconds since 1970-01-01.

- int os_daemonize (const char *pid_file)

  Run in the background (detach from the controlling terminal).

- void os_daemonize_terminate (const char *pid_file)

  Stop running in the background (remove pid file).

- int os_get_random (unsigned char *buf, size_t len)

  Get cryptographically strong pseudo random data.

- unsigned long os_random (void)

  Get pseudo random value (not necessarily very strong).

- char * os_rel2abs_path (const char *rel_path)

  Get an absolute path for a file.

- int os_program_init (void)

  Program initialization (called at start).

- void os_program_deinit (void)

  Program deinitialization (called just before exit).

- int os_setenv (const char *name, const char *value, int overwrite)

  Set environment variable.

- int os_unsetenv (const char *name)

  Delete environent variable.

- char * os_readfile (const char *name, size_t *len)

  Read a file to an allocated memory buffer.

- void * os_zalloc (size_t size)

  Allocate and zero memory.

- size_t os_strlcpy (char *dest, const char *src, size_t siz)

  Copy a string with size bound and NUL-termination.

15.330.1 Detailed Description

wpa_supplicant/hostapd / OS specific functions

Copyright

15.330.2 Define Documentation

15.330.2.1 #define os_time_before(a, b)

Value:

```
((a)->sec < (b)->sec || \
         ((a)->sec == (b)->sec && (a)->usec < (b)->usec))
```

15.330.2.2 #define os_time_sub(a, b, res)

Value:

```
do { \
        (res)->sec = (a)->sec - (b)->sec; \
        (res)->usec = (a)->usec - (b)->usec; \
        if ((res)->usec < 0) { \
                (res)->sec--; \
                (res)->usec += 1000000; \
        } \
} while (0)
```

15.330.3 Function Documentation

15.330.3.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

Returns:

0 on success, -1 on failure

15.330.3.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

15.330.3.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

Parameters:

buf Buffer for pseudo random data

len Length of the buffer

Returns:

0 on success, -1 on failure

15.330.3.4 int os_get_time (struct os_time * t)

Get current time (sec, usec).

Parameters:

t Pointer to buffer for the time

Returns:

0 on success, -1 on failure

15.330.3.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * t)

Convert broken-down time into seconds since 1970-01-01.

Parameters:

year Four digit year

month Month (1 .. 12)

day Day of month (1 .. 31)

hour Hour (0 .. 23)

min Minute (0 .. 59)

sec Second (0 .. 60)

t Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

Returns:

0 on success, -1 on failure

Note: The result is in seconds from Epoch, i.e., in UTC, not in local time which is used by POSIX mktime().

15.330.3.6 void os_program_deinit (void)

Program deinitialization (called just before exit). This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

15.330.3.7 int os_program_init (void)

Program initialization (called at start).

Returns:

0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

15.330.3.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

Returns:

Pseudo random value

15.330.3.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

Parameters:

name Name of the file to read

len For returning the length of the allocated buffer

Returns:

Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

15.330.3.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

Parameters:

rel_path Relative path to a file

Returns:

Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

15.330.3.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

Parameters:

name Name of the variable

value Value to set to the variable

overwrite Whether existing variable should be overwritten

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.330.3.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

Parameters:

sec Number of seconds to sleep

usec Number of microseconds to sleep

15.330.3.13 size_t os_strlcpy (char * *dest*, const char * *src*, size_t *siz*)

Copy a string with size bound and NUL-termination.

Parameters:

dest Destination

src Source

siz Size of the target buffer

Returns:

Total length of the target string (length of src) (not including NUL-termination)

This function matches in behavior with the strlcpy(3) function in OpenBSD.

15.330.3.14 int os_unsetenv (const char * *name*)

Delete environent variable.

Parameters:

name Name of the variable

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.330.3.15 void* os_zalloc (size_t *size*)

Allocate and zero memory.

Parameters:

size Number of bytes to allocate

Returns:

Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

15.331 src/utils/os_internal.c File Reference

wpa_supplicant/hostapd / Internal implementation of OS specific functions `#include "includes.h"` `#include "os.h"`

Functions

- void os_sleep (os_time_t sec, os_time_t usec)
  Sleep (sec, usec).
- int os_get_time (struct os_time *t)
  Get current time (sec, usec).
- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)
  Convert broken-down time into seconds since 1970-01-01.
- int os_daemonize (const char *pid_file)
  Run in the background (detach from the controlling terminal).
- void os_daemonize_terminate (const char *pid_file)
  Stop running in the background (remove pid file).
- int os_get_random (unsigned char *buf, size_t len)
  Get cryptographically strong pseudo random data.
- unsigned long os_random (void)
  Get pseudo random value (not necessarily very strong).
- char * os_rel2abs_path (const char *rel_path)
  Get an absolute path for a file.
- int os_program_init (void)
  Program initialization (called at start).
- void os_program_deinit (void)
  Program deinitialization (called just before exit).
- int os_setenv (const char *name, const char *value, int overwrite)
  Set environment variable.
- int os_unsetenv (const char *name)
  Delete environent variable.
- char * os_readfile (const char *name, size_t *len)
  Read a file to an allocated memory buffer.
- void * os_zalloc (size_t size)
  Allocate and zero memory.

- void * **os_malloc** (size_t size)
- void * **os_realloc** (void *ptr, size_t size)
- void **os_free** (void *ptr)
- void * **os_memcpy** (void *dest, const void *src, size_t n)
- void * **os_memmove** (void *dest, const void *src, size_t n)
- void * **os_memset** (void *s, int c, size_t n)
- int **os_memcmp** (const void *s1, const void *s2, size_t n)
- char * **os_strdup** (const char *s)
- size_t **os_strlen** (const char *s)
- int **os_strcasecmp** (const char *s1, const char *s2)
- int **os_strncasecmp** (const char *s1, const char *s2, size_t n)
- char * **os_strchr** (const char *s, int c)
- char * **os_strrchr** (const char *s, int c)
- int **os_strcmp** (const char *s1, const char *s2)
- int **os_strncmp** (const char *s1, const char *s2, size_t n)
- char * **os_strncpy** (char *dest, const char *src, size_t n)
- size_t os_strlcpy (char *dest, const char *src, size_t siz)

  Copy a string with size bound and NUL-termination.

- char * **os_strstr** (const char *haystack, const char *needle)
- int **os_snprintf** (char *str, size_t size, const char *format,...)

15.331.1 Detailed Description

wpa_supplicant/hostapd / Internal implementation of OS specific functions

Copyright

Copyright (c) 2005-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file is an example of operating system specific wrapper functions. This version implements many of the functions internally, so it can be used to fill in missing functions from the target system C libraries.

Some of the functions are using standard C library calls in order to keep this file in working condition to allow the functions to be tested on a Linux target. Please note that OS_NO_C_LIB_DEFINES needs to be defined for this file to work correctly. Note that these implementations are only examples and are not optimized for speed.

15.331.2 Function Documentation

15.331.2.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

Returns:

0 on success, -1 on failure

15.331.2.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

15.331.2.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

Parameters:

buf Buffer for pseudo random data

len Length of the buffer

Returns:

0 on success, -1 on failure

15.331.2.4 int os_get_time (struct os_time * *t*)

Get current time (sec, usec).

Parameters:

t Pointer to buffer for the time

Returns:

0 on success, -1 on failure

15.331.2.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * *t*)

Convert broken-down time into seconds since 1970-01-01.

Parameters:

year Four digit year

month Month (1 .. 12)

day Day of month (1 .. 31)

hour Hour (0 .. 23)

min Minute (0 .. 59)

sec Second (0 .. 60)

t Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

Returns:

0 on success, -1 on failure

Note: The result is in seconds from Epoch, i.e., in UTC, not in local time which is used by POSIX mktime().

15.331.2.6 void os_program_deinit (void)

Program deinitialization (called just before exit). This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

15.331.2.7 int os_program_init (void)

Program initialization (called at start).

Returns:

0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

15.331.2.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

Returns:

Pseudo random value

15.331.2.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

Parameters:

name Name of the file to read

len For returning the length of the allocated buffer

Returns:

Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

15.331.2.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

Parameters:

rel_path Relative path to a file

Returns:

Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

15.331.2.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

Parameters:

name Name of the variable

value Value to set to the variable

overwrite Whether existing variable should be overwritten

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.331.2.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

Parameters:

sec Number of seconds to sleep

usec Number of microseconds to sleep

15.331.2.13 size_t os_strlcpy (char * *dest*, const char * *src*, size_t *siz*)

Copy a string with size bound and NUL-termination.

Parameters:

dest Destination

src Source

siz Size of the target buffer

Returns:

Total length of the target string (length of src) (not including NUL-termination)

This function matches in behavior with the strlcpy(3) function in OpenBSD.

15.331.2.14 int os_unsetenv (const char * *name*)

Delete environent variable.

Parameters:

name Name of the variable

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.331.2.15 void* os_zalloc (size_t *size*)

Allocate and zero memory.

Parameters:

size Number of bytes to allocate

Returns:

Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

15.332 src/utils/os_none.c File Reference

wpa_supplicant/hostapd / Empty OS specific functions `#include "includes.h"`

`#include "os.h"`

Functions

- void os_sleep (os_time_t sec, os_time_t usec)
  Sleep (sec, usec).
- int os_get_time (struct os_time *t)
  Get current time (sec, usec).
- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)
  Convert broken-down time into seconds since 1970-01-01.
- int os_daemonize (const char *pid_file)
  Run in the background (detach from the controlling terminal).
- void os_daemonize_terminate (const char *pid_file)
  Stop running in the background (remove pid file).
- int os_get_random (unsigned char *buf, size_t len)
  Get cryptographically strong pseudo random data.
- unsigned long os_random (void)
  Get pseudo random value (not necessarily very strong).
- char * os_rel2abs_path (const char *rel_path)
  Get an absolute path for a file.
- int os_program_init (void)
  Program initialization (called at start).
- void os_program_deinit (void)
  Program deinitialization (called just before exit).
- int os_setenv (const char *name, const char *value, int overwrite)
  Set environment variable.
- int os_unsetenv (const char *name)
  Delete environent variable.
- char * os_readfile (const char *name, size_t *len)
  Read a file to an allocated memory buffer.
- void * os_zalloc (size_t size)
  Allocate and zero memory.

15.332.1 Detailed Description

wpa_supplicant/hostapd / Empty OS specific functions

Copyright

Copyright (c) 2005-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file can be used as a starting point when adding a new OS target. The functions here do not really work as-is since they are just empty or only return an error value. os_internal.c can be used as another starting point or reference since it has example implementation of many of these functions.

15.332.2 Function Documentation

15.332.2.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

Returns:

0 on success, -1 on failure

15.332.2.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

15.332.2.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

Parameters:

buf Buffer for pseudo random data

len Length of the buffer

Returns:

0 on success, -1 on failure

15.332.2.4 int os_get_time (struct os_time * t)

Get current time (sec, usec).

Parameters:

t Pointer to buffer for the time

Returns:

0 on success, -1 on failure

15.332.2.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * t)

Convert broken-down time into seconds since 1970-01-01.

Parameters:

year Four digit year

month Month (1 .. 12)

day Day of month (1 .. 31)

hour Hour (0 .. 23)

min Minute (0 .. 59)

sec Second (0 .. 60)

t Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

Returns:

0 on success, -1 on failure

Note: The result is in seconds from Epoch, i.e., in UTC, not in local time which is used by POSIX mktime().

15.332.2.6 void os_program_deinit (void)

Program deinitialization (called just before exit). This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

15.332.2.7 int os_program_init (void)

Program initialization (called at start).

Returns:

0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

15.332.2.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

Returns:

Pseudo random value

15.332.2.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

Parameters:

name Name of the file to read

len For returning the length of the allocated buffer

Returns:

Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

15.332.2.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

Parameters:

rel_path Relative path to a file

Returns:

Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

15.332.2.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

Parameters:

name Name of the variable

value Value to set to the variable

overwrite Whether existing variable should be overwritten

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.332.2.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

Parameters:

sec Number of seconds to sleep

usec Number of microseconds to sleep

15.332.2.13 int os_unsetenv (const char * *name*)

Delete environent variable.

Parameters:

name Name of the variable

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.332.2.14 void* os_zalloc (size_t *size*)

Allocate and zero memory.

Parameters:

size Number of bytes to allocate

Returns:

Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

15.333 src/utils/os_unix.c File Reference

wpa_supplicant/hostapd / OS specific functions for UNIX/POSIX systems `#include "includes.h"` `#include "os.h"`

Defines

- #define **os_daemon** daemon

Functions

- void os_sleep (os_time_t sec, os_time_t usec)
  Sleep (sec, usec).
- int os_get_time (struct os_time *t)
  Get current time (sec, usec).
- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)
  Convert broken-down time into seconds since 1970-01-01.
- int os_daemonize (const char *pid_file)
  Run in the background (detach from the controlling terminal).
- void os_daemonize_terminate (const char *pid_file)
  Stop running in the background (remove pid file).
- int os_get_random (unsigned char *buf, size_t len)
  Get cryptographically strong pseudo random data.
- unsigned long os_random (void)
  Get pseudo random value (not necessarily very strong).
- char * os_rel2abs_path (const char *rel_path)
  Get an absolute path for a file.
- int os_program_init (void)
  Program initialization (called at start).
- void os_program_deinit (void)
  Program deinitialization (called just before exit).
- int os_setenv (const char *name, const char *value, int overwrite)
  Set environment variable.
- int os_unsetenv (const char *name)
  Delete environent variable.
- char * os_readfile (const char *name, size_t *len)

Read a file to an allocated memory buffer.

- void * os_zalloc (size_t size)
  Allocate and zero memory.
- size_t os_strlcpy (char *dest, const char *src, size_t siz)
  Copy a string with size bound and NUL-termination.

15.333.1 Detailed Description

wpa_supplicant/hostapd / OS specific functions for UNIX/POSIX systems

Copyright

Copyright (c) 2005-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.333.2 Function Documentation

15.333.2.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

Returns:

0 on success, -1 on failure

15.333.2.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

15.333.2.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

Parameters:

buf Buffer for pseudo random data

len Length of the buffer

Returns:

0 on success, -1 on failure

15.333.2.4 int os_get_time (struct os_time * *t*)

Get current time (sec, usec).

Parameters:

t Pointer to buffer for the time

Returns:

0 on success, -1 on failure

15.333.2.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * *t*)

Convert broken-down time into seconds since 1970-01-01.

Parameters:

year Four digit year
month Month (1 .. 12)
day Day of month (1 .. 31)
hour Hour (0 .. 23)
min Minute (0 .. 59)
sec Second (0 .. 60)
t Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

Returns:

0 on success, -1 on failure

Note: The result is in seconds from Epoch, i.e., in UTC, not in local time which is used by POSIX mktime().

15.333.2.6 void os_program_deinit (void)

Program deinitialization (called just before exit). This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

15.333.2.7 int os_program_init (void)

Program initialization (called at start).

Returns:

0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

15.333.2.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

Returns:

Pseudo random value

15.333.2.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

Parameters:

name Name of the file to read

len For returning the length of the allocated buffer

Returns:

Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

15.333.2.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

Parameters:

rel_path Relative path to a file

Returns:

Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

15.333.2.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

Parameters:

name Name of the variable

value Value to set to the variable

overwrite Whether existing variable should be overwritten

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.333.2.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

Parameters:

sec Number of seconds to sleep

usec Number of microseconds to sleep

15.333.2.13 size_t os_strlcpy (char * *dest*, const char * *src*, size_t *siz*)

Copy a string with size bound and NUL-termination.

Parameters:

dest Destination

src Source

siz Size of the target buffer

Returns:

Total length of the target string (length of src) (not including NUL-termination)

This function matches in behavior with the strlcpy(3) function in OpenBSD.

15.333.2.14 int os_unsetenv (const char * *name*)

Delete environent variable.

Parameters:

name Name of the variable

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.333.2.15 void* os_zalloc (size_t *size*)

Allocate and zero memory.

Parameters:

size Number of bytes to allocate

Returns:

Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

15.334 src/utils/os_win32.c File Reference

wpa_supplicant/hostapd / OS specific functions for Win32 systems `#include "includes.h"`

```
#include <winsock2.h>
#include <wincrypt.h>
#include "os.h"
```

Defines

- #define **EPOCHFILETIME** (116444736000000000ULL)

Functions

- void os_sleep (os_time_t sec, os_time_t usec)
  Sleep (sec, usec).
- int os_get_time (struct os_time *t)
  Get current time (sec, usec).
- int os_mktime (int year, int month, int day, int hour, int min, int sec, os_time_t *t)
  Convert broken-down time into seconds since 1970-01-01.
- int os_daemonize (const char *pid_file)
  Run in the background (detach from the controlling terminal).
- void os_daemonize_terminate (const char *pid_file)
  Stop running in the background (remove pid file).
- int os_get_random (unsigned char *buf, size_t len)
  Get cryptographically strong pseudo random data.
- unsigned long os_random (void)
  Get pseudo random value (not necessarily very strong).
- char * os_rel2abs_path (const char *rel_path)
  Get an absolute path for a file.
- int os_program_init (void)
  Program initialization (called at start).
- void os_program_deinit (void)
  Program deinitialization (called just before exit).
- int os_setenv (const char *name, const char *value, int overwrite)
  Set environment variable.
- int os_unsetenv (const char *name)
  Delete environent variable.

- char * os_readfile (const char *name, size_t *len)

  Read a file to an allocated memory buffer.

- void * os_zalloc (size_t size)

  Allocate and zero memory.

- size_t os_strlcpy (char *dest, const char *src, size_t siz)

  Copy a string with size bound and NUL-termination.

15.334.1 Detailed Description

wpa_supplicant/hostapd / OS specific functions for Win32 systems

Copyright

Copyright (c) 2005-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.334.2 Function Documentation

15.334.2.1 int os_daemonize (const char * *pid_file*)

Run in the background (detach from the controlling terminal).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

Returns:

0 on success, -1 on failure

15.334.2.2 void os_daemonize_terminate (const char * *pid_file*)

Stop running in the background (remove pid file).

Parameters:

pid_file File name to write the process ID to or NULL to skip this

15.334.2.3 int os_get_random (unsigned char * *buf*, size_t *len*)

Get cryptographically strong pseudo random data.

Parameters:

buf Buffer for pseudo random data

len Length of the buffer

Returns:

0 on success, -1 on failure

15.334.2.4 int os_get_time (struct os_time * *t*)

Get current time (sec, usec).

Parameters:

t Pointer to buffer for the time

Returns:

0 on success, -1 on failure

15.334.2.5 int os_mktime (int *year*, int *month*, int *day*, int *hour*, int *min*, int *sec*, os_time_t * *t*)

Convert broken-down time into seconds since 1970-01-01.

Parameters:

year Four digit year

month Month (1 .. 12)

day Day of month (1 .. 31)

hour Hour (0 .. 23)

min Minute (0 .. 59)

sec Second (0 .. 60)

t Buffer for returning calendar time representation (seconds since 1970-01-01 00:00:00)

Returns:

0 on success, -1 on failure

Note: The result is in seconds from Epoch, i.e., in UTC, not in local time which is used by POSIX mktime().

15.334.2.6 void os_program_deinit (void)

Program deinitialization (called just before exit). This function is called just before a program exists. If there are any OS specific processing, e.g., freeing resourced allocated in os_program_init(), it should be done here. It is also acceptable for this function to do nothing.

15.334.2.7 int os_program_init (void)

Program initialization (called at start).

Returns:

0 on success, -1 on failure

This function is called when a programs starts. If there are any OS specific processing that is needed, it can be placed here. It is also acceptable to just return 0 if not special processing is needed.

15.334.2.8 unsigned long os_random (void)

Get pseudo random value (not necessarily very strong).

Returns:

Pseudo random value

15.334.2.9 char* os_readfile (const char * *name*, size_t * *len*)

Read a file to an allocated memory buffer.

Parameters:

name Name of the file to read

len For returning the length of the allocated buffer

Returns:

Pointer to the allocated buffer or NULL on failure

This function allocates memory and reads the given file to this buffer. Both binary and text files can be read with this function. The caller is responsible for freeing the returned buffer with os_free().

15.334.2.10 char* os_rel2abs_path (const char * *rel_path*)

Get an absolute path for a file.

Parameters:

rel_path Relative path to a file

Returns:

Absolute path for the file or NULL on failure

This function tries to convert a relative path of a file to an absolute path in order for the file to be found even if current working directory has changed. The returned value is allocated and caller is responsible for freeing it. It is acceptable to just return the same path in an allocated buffer, e.g., return strdup(rel_path). This function is only used to find configuration files when os_daemonize() may have changed the current working directory and relative path would be pointing to a different location.

15.334.2.11 int os_setenv (const char * *name*, const char * *value*, int *overwrite*)

Set environment variable.

Parameters:

name Name of the variable

value Value to set to the variable

overwrite Whether existing variable should be overwritten

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.334.2.12 void os_sleep (os_time_t *sec*, os_time_t *usec*)

Sleep (sec, usec).

Parameters:

sec Number of seconds to sleep

usec Number of microseconds to sleep

15.334.2.13 size_t os_strlcpy (char * *dest*, const char * *src*, size_t *siz*)

Copy a string with size bound and NUL-termination.

Parameters:

dest Destination

src Source

siz Size of the target buffer

Returns:

Total length of the target string (length of src) (not including NUL-termination)

This function matches in behavior with the strlcpy(3) function in OpenBSD.

15.334.2.14 int os_unsetenv (const char * *name*)

Delete environent variable.

Parameters:

name Name of the variable

Returns:

0 on success, -1 on error

This function is only used for wpa_cli action scripts. OS wrapper does not need to implement this if such functionality is not needed.

15.334.2.15 void* os_zalloc (size_t *size*)

Allocate and zero memory.

Parameters:

size Number of bytes to allocate

Returns:

Pointer to allocated and zeroed memory or NULL on failure

Caller is responsible for freeing the returned buffer with os_free().

15.335 src/utils/pcsc_funcs.c File Reference

WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM. `#include "includes.h"`

`#include <winscard.h>`

`#include "common.h"`

`#include "pcsc_funcs.h"`

Data Structures

- struct scard_data

Defines

- #define **SIM_CMD_SELECT** 0xa0, 0xa4, 0x00, 0x00, 0x02
- #define **SIM_CMD_RUN_GSM_ALG** 0xa0, 0x88, 0x00, 0x00, 0x10
- #define **SIM_CMD_GET_RESPONSE** 0xa0, 0xc0, 0x00, 0x00
- #define **SIM_CMD_READ_BIN** 0xa0, 0xb0, 0x00, 0x00
- #define **SIM_CMD_READ_RECORD** 0xa0, 0xb2, 0x00, 0x00
- #define **SIM_CMD_VERIFY_CHV1** 0xa0, 0x20, 0x00, 0x01, 0x08
- #define **USIM_CLA** 0x00
- #define **USIM_CMD_RUN_UMTS_ALG** 0x00, 0x88, 0x00, 0x81, 0x22
- #define **USIM_CMD_GET_RESPONSE** 0x00, 0xc0, 0x00, 0x00
- #define **SIM_RECORD_MODE_ABSOLUTE** 0x04
- #define **USIM_FSP_TEMPL_TAG** 0x62
- #define **USIM_TLV_FILE_DESC** 0x82
- #define **USIM_TLV_FILE_ID** 0x83
- #define **USIM_TLV_DF_NAME** 0x84
- #define **USIM_TLV_PROPR_INFO** 0xA5
- #define **USIM_TLV_LIFE_CYCLE_STATUS** 0x8A
- #define **USIM_TLV_FILE_SIZE** 0x80
- #define **USIM_TLV_TOTAL_FILE_SIZE** 0x81
- #define **USIM_TLV_PIN_STATUS_TEMPLATE** 0xC6
- #define **USIM_TLV_SHORT_FILE_ID** 0x88
- #define **USIM_PS_DO_TAG** 0x90
- #define **AKA_RAND_LEN** 16
- #define **AKA_AUTN_LEN** 16
- #define **AKA_AUTS_LEN** 14
- #define **RES_MAX_LEN** 16
- #define **IK_LEN** 16
- #define **CK_LEN** 16
- #define **mingw_load_symbols**() 0
- #define **mingw_unload_symbols**() do { } while (0)

Enumerations

- enum **sim_types** { **SCARD_GSM_SIM**, **SCARD_USIM** }

Functions

- struct 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.

15.335.1 Detailed Description

WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM.

Copyright

Copyright (c) 2004-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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.

15.335.2 Function Documentation

15.335.2.1 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().

15.335.2.2 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().

15.335.2.3 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.

15.335.2.4 struct scard_data* scard_init (scard_sim_type *sim_type*) [read]

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.

15.335.2.5 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

15.335.2.6 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.

15.336 src/utils/pcsc_funcs.h File Reference

WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM.

Defines

- #define **SCARD_FILE_MF** 0x3F00
- #define **SCARD_FILE_GSM_DF** 0x7F20
- #define **SCARD_FILE_UMTS_DF** 0x7F50
- #define **SCARD_FILE_GSM_EF_IMSI** 0x6F07
- #define **SCARD_FILE_EF_DIR** 0x2F00
- #define **SCARD_FILE_EF_ICCID** 0x2FE2
- #define **SCARD_FILE_EF_CK** 0x6FE1
- #define **SCARD_FILE_EF_IK** 0x6FE2
- #define **SCARD_CHV1_OFFSET** 13
- #define **SCARD_CHV1_FLAG** 0x80
- #define **scard_init**(s) NULL
- #define **scard_deinit**(s) do { } while (0)
- #define **scard_set_pin**(s, p) -1
- #define **scard_get_imsi**(s, i, l) -1
- #define **scard_gsm_auth**(s, r, s2, k) -1
- #define **scard_umts_auth**(s, r, a, r2, rl, i, c, a2) -1

Enumerations

- enum **scard_sim_type** { **SCARD_GSM_SIM_ONLY**, **SCARD_USIM_ONLY**, **SCARD_TRY_BOTH** }

15.336.1 Detailed Description

WPA Supplicant / PC/SC smartcard interface for USIM, GSM SIM.

Copyright

15.337 src/utils/radiotap.c File Reference

Radiotap parser. `#include "includes.h"`

`#include "common.h"`

`#include "radiotap_iter.h"`

Defines

- #define **le16_to_cpu** le_to_host16
- #define **le32_to_cpu** le_to_host32
- #define **__le32** uint32_t
- #define **ulong** unsigned long
- #define **unlikely**(cond) (cond)
- #define **get_unaligned**(p)

Functions

- int ieee80211_radiotap_iterator_init (struct ieee80211_radiotap_iterator *iterator, struct ieee80211_radiotap_header *radiotap_header, int max_length)

  radiotap parser iterator initialization

- int ieee80211_radiotap_iterator_next (struct ieee80211_radiotap_iterator *iterator)

  return next radiotap parser iterator arg

15.337.1 Detailed Description

Radiotap parser.

Copyright

Copyright 2007 Andy Green <andy@warmcat.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.

Modified for userspace by Johannes Berg <johannes@sipsolutions.net> I only modified some things on top to ease syncing should bugs be found.

15.337.2 Define Documentation

15.337.2.1 #define get_unaligned(p)

Value:

```
({                                                              \
        struct packed_dummy_struct {                            \
                typeof(*(p)) __val;                             \
        } __attribute__((packed)) *__ptr = (void *) (p);        \
                                                                \
        __ptr->__val;                                           \
})
```

15.337.3 Function Documentation

15.337.3.1 int ieee80211_radiotap_iterator_init (struct ieee80211_radiotap_iterator * *iterator*, struct ieee80211_radiotap_header * *radiotap_header*, int *max_length*)

radiotap parser iterator initialization

Parameters:

iterator radiotap_iterator to initialize

radiotap_header radiotap header to parse

max_length total length we can parse into (eg, whole packet length)

Returns:

0 or a negative error code if there is a problem.

This function initializes an opaque iterator struct which can then be passed to ieee80211_radiotap_iterator_next() to visit every radiotap argument which is present in the header. It knows about extended present headers and handles them.

How to use: call __ieee80211_radiotap_iterator_init() to init a semi-opaque iterator struct ieee80211_radiotap_iterator (no need to init the struct beforehand) checking for a good 0 return code. Then loop calling __ieee80211_radiotap_iterator_next()... it returns either 0, -ENOENT if there are no more args to parse, or -EINVAL if there is a problem. The iterator's member points to the start of the argument associated with the current argument index that is present, which can be found in the iterator's member. This arg index corresponds to the IEEE80211_RADIOTAP_... defines.

Radiotap header length: You can find the CPU-endian total radiotap header length in iterator->max_length after executing ieee80211_radiotap_iterator_init() successfully.

Alignment Gotcha: You must take care when dereferencing iterator.this_arg for multibyte types... the pointer is not aligned. Use get_unaligned((type *)iterator.this_arg) to dereference iterator.this_arg for type "type" safely on all arches.

Example code: See Documentation/networking/radiotap-headers.txt

15.337.3.2 int ieee80211_radiotap_iterator_next (struct ieee80211_radiotap_iterator * *iterator*)

return next radiotap parser iterator arg

Parameters:

iterator radiotap_iterator to move to next arg (if any)

Returns:

0 if there is an argument to handle, -ENOENT if there are no more args or -EINVAL if there is something else wrong.

This function provides the next radiotap arg index (IEEE80211_RADIOTAP_∗) in and sets to point to the payload for the field. It takes care of alignment handling and extended present fields. can be changed by the caller (eg, incremented to move inside a compound argument like IEEE80211_RADIOTAP_CHANNEL). The args pointed to are in little-endian format whatever the endianess of your CPU.

Alignment Gotcha: You must take care when dereferencing iterator.this_arg for multibyte types... the pointer is not aligned. Use get_unaligned((type ∗)iterator.this_arg) to dereference iterator.this_arg for type "type" safely on all arches.

15.338 src/utils/state_machine.h File Reference

wpa_supplicant/hostapd - State machine definitions

Defines

- #define SM_STATE(machine, state)
  Declaration of a state machine function.
- #define SM_ENTRY(machine, state)
  State machine function entry point.
- #define SM_ENTRY_M(machine, _state, data)
  State machine function entry point for state machine group.
- #define SM_ENTRY_MA(machine, _state, data)
  State machine function entry point for state machine group.
- #define SM_ENTER(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 0)
  Enter a new state machine state.
- #define SM_ENTER_GLOBAL(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 1)
  Enter a new state machine state based on global rule.
- #define SM_STEP(machine) static void sm_ ## machine ## _Step(STATE_MACHINE_DATA *sm)
  Declaration of a state machine step function.
- #define SM_STEP_RUN(machine) sm_ ## machine ## _Step(sm)
  Call the state machine step function.

15.338.1 Detailed Description

wpa_supplicant/hostapd - State machine definitions

Copyright

Copyright (c) 2002-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file includes a set of pre-processor macros that can be used to implement a state machine. In addition to including this header file, each file implementing a state machine must define STATE_MACHINE_DATA to be the data structure including state variables (enum machine_state, Boolean changed), and STATE_MACHINE_DEBUG_PREFIX to be a string that is used as a prefix for all debug messages. If SM_ENTRY_MA macro is used to define a group of state machines with shared data structure, STATE_MACHINE_ADDR needs to be defined to point to the MAC address used in debug output. SM_ENTRY_M macro can be used to define similar group of state machines without this additional debug info.

15.338.2 Define Documentation

15.338.2.1 #define SM_ENTER(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 0)

Enter a new state machine state.

Parameters:

machine State machine name

state State machine state

This macro expands to a function call to a state machine function defined with SM_STATE macro. SM_-ENTER is used in a state machine step function to move the state machine to a new state.

15.338.2.2 #define SM_ENTER_GLOBAL(machine, state) sm_ ## machine ## _ ## state ## _Enter(sm, 1)

Enter a new state machine state based on global rule.

Parameters:

machine State machine name

state State machine state

This macro is like SM_ENTER, but this is used when entering a new state based on a global (not specific to any particular state) rule. A separate macro is used to avoid unwanted debug message floods when the same global rule is forcing a state machine to remain in on state.

15.338.2.3 #define SM_ENTRY(machine, state)

Value:

```
if (!global || sm->machine ## _state != machine ## _ ## state) { \
        sm->changed = TRUE; \
        wpa_printf(MSG_DEBUG, STATE_MACHINE_DEBUG_PREFIX ": " #machine \
                   " entering state " #state); \
} \
sm->machine ## _state = machine ## _ ## state;
```

State machine function entry point.

Parameters:

machine State machine name

state State machine state

This macro is used inside each state machine function declared with SM_STATE. SM_ENTRY should be in the beginning of the function body, but after declaration of possible local variables. This macro prints debug information about state transition and update the state machine state.

15.338.2.4 #define SM_ENTRY_M(machine, _state, data)

Value:

```
if (!global || sm->data ## _ ## state != machine ## _ ## _state) { \
        sm->changed = TRUE; \
        wpa_printf(MSG_DEBUG, STATE_MACHINE_DEBUG_PREFIX ": " \
                   #machine " entering state " #_state); \
} \
sm->data ## _ ## state = machine ## _ ## _state;
```

State machine function entry point for state machine group.

Parameters:

machine State machine name

_state State machine state

data State variable prefix (full variable: prefix_state)

This macro is like SM_ENTRY, but for state machine groups that use a shared data structure for more than one state machine. Both machine and prefix parameters are set to "sub-state machine" name. prefix is used to allow more than one state variable to be stored in the same data structure.

15.338.2.5 #define SM_ENTRY_MA(machine, _state, data)

Value:

```
if (!global || sm->data ## _ ## state != machine ## _ ## _state) { \
        sm->changed = TRUE; \
        wpa_printf(MSG_DEBUG, STATE_MACHINE_DEBUG_PREFIX ": " MACSTR " " \
                   #machine " entering state " #_state, \
                   MAC2STR(STATE_MACHINE_ADDR)); \
} \
sm->data ## _ ## state = machine ## _ ## _state;
```

State machine function entry point for state machine group.

Parameters:

machine State machine name

_state State machine state

data State variable prefix (full variable: prefix_state)

This macro is like SM_ENTRY_M, but a MAC address is included in debug output. STATE_MACHINE_-ADDR has to be defined to point to the MAC address to be included in debug.

15.338.2.6 #define SM_STATE(machine, state)

Value:

```
static void sm_ ## machine ## _ ## state ## _Enter(STATE_MACHINE_DATA *sm, \
        int global)
```

Declaration of a state machine function.

Parameters:

machine State machine name

state State machine state

This macro is used to declare a state machine function. It is used in place of a C function definition to declare functions to be run when the state is entered by calling SM_ENTER or SM_ENTER_GLOBAL.

15.338.2.7 #define SM_STEP(machine) static void sm_ ## machine ## _Step(STATE_MACHINE_DATA *sm)

Declaration of a state machine step function.

Parameters:

machine State machine name

This macro is used to declare a state machine step function. It is used in place of a C function definition to declare a function that is used to move state machine to a new state based on state variables. This function uses SM_ENTER and SM_ENTER_GLOBAL macros to enter new state.

15.338.2.8 #define SM_STEP_RUN(machine) sm_ ## machine ## _Step(sm)

Call the state machine step function.

Parameters:

machine State machine name

This macro expands to a function call to a state machine step function defined with SM_STEP macro.

15.339 src/utils/uuid.c File Reference

Universally Unique IDentifier (UUID). `#include "includes.h"`

`#include "common.h"`

`#include "crypto.h"`

`#include "sha1.h"`

`#include "uuid.h"`

Functions

- int **uuid_str2bin** (const char *str, u8 *bin)
- int **uuid_bin2str** (const u8 *bin, char *str, size_t max_len)
- int **is_nil_uuid** (const u8 *uuid)
- void **uuid_gen_mac_addr** (const u8 *mac_addr, u8 *uuid)

15.339.1 Detailed Description

Universally Unique IDentifier (UUID).

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.340 src/utils/uuid.h File Reference

Universally Unique IDentifier (UUID).

Defines

- #define **UUID_LEN** 16

Functions

- int **uuid_str2bin** (const char *str, u8 *bin)
- int **uuid_bin2str** (const u8 *bin, char *str, size_t max_len)
- int **is_nil_uuid** (const u8 *uuid)
- void **uuid_gen_mac_addr** (const u8 *mac_addr, u8 *uuid)

15.340.1 Detailed Description

Universally Unique IDentifier (UUID).

Copyright

15.341 src/utils/wpa_debug.c File Reference

wpa_supplicant/hostapd / Debug prints `#include "includes.h"`

`#include "common.h"`

Functions

- void wpa_debug_print_timestamp (void)

  Print timestamp for debug output.

- void wpa_printf (int level, char *fmt,...)

  conditional printf

- void wpa_hexdump (int level, const char *title, const u8 *buf, size_t len)

  conditional hex dump

- void wpa_hexdump_key (int level, const char *title, const u8 *buf, size_t len)

  conditional hex dump, hide keys

- void wpa_hexdump_ascii (int level, const char *title, const u8 *buf, size_t len)

  conditional hex dump

- void wpa_hexdump_ascii_key (int level, const char *title, const u8 *buf, size_t len)

  conditional hex dump, hide keys

- int **wpa_debug_open_file** (const char *path)
- void **wpa_debug_close_file** (void)
- void wpa_msg_register_cb (wpa_msg_cb_func func)

  Register callback function for wpa_msg() messages.

- void **wpa_msg** (void *ctx, int level, char *fmt,...)
- void **wpa_msg_ctrl** (void *ctx, int level, char *fmt,...)
- void hostapd_logger_register_cb (hostapd_logger_cb_func func)

  Register callback function for hostapd_logger().

- void **hostapd_logger** (void *ctx, const u8 *addr, unsigned int module, int level, const char *fmt,...)

Variables

- int **wpa_debug_level** = MSG_INFO
- int **wpa_debug_show_keys** = 0
- int **wpa_debug_timestamp** = 0

15.341.1 Detailed Description

wpa_supplicant/hostapd / Debug prints

Copyright

15.341.2 Function Documentation

15.341.2.1 void hostapd_logger_register_cb (hostapd_logger_cb_func *func*)

Register callback function for hostapd_logger().

Parameters:

func Callback function (NULL to unregister)

15.341.2.2 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.

15.341.2.3 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.

15.341.2.4 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.

15.341.2.5 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.

15.341.2.6 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.

15.341.2.7 void wpa_msg_register_cb (wpa_msg_cb_func *func*)

Register callback function for wpa_msg() messages.

Parameters:

func Callback function (NULL to unregister)

15.341.2.8 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.

15.342 src/utils/wpa_debug.h File Reference

wpa_supplicant/hostapd / Debug prints `#include "wpabuf.h"`

Defines

- #define **HOSTAPD_MODULE_IEEE80211** 0x00000001
- #define **HOSTAPD_MODULE_IEEE8021X** 0x00000002
- #define **HOSTAPD_MODULE_RADIUS** 0x00000004
- #define **HOSTAPD_MODULE_WPA** 0x00000008
- #define **HOSTAPD_MODULE_DRIVER** 0x00000010
- #define **HOSTAPD_MODULE_IAPP** 0x00000020
- #define **HOSTAPD_MODULE_MLME** 0x00000040
- #define **WPA_ASSERT**(a) do { } while (0)

Enumerations

- enum {
  MSG_MSGDUMP, **MSG_DEBUG**, **MSG_INFO**, **MSG_WARNING**,
  MSG_ERROR }
- enum **hostapd_logger_level** {
  HOSTAPD_LEVEL_DEBUG_VERBOSE = 0, **HOSTAPD_LEVEL_DEBUG** = 1, **HOSTAPD_LEVEL_INFO** = 2, **HOSTAPD_LEVEL_NOTICE** = 3,
  HOSTAPD_LEVEL_WARNING = 4 }

Functions

- int **wpa_debug_open_file** (const char *path)
- void **wpa_debug_close_file** (void)
- void wpa_debug_print_timestamp (void)
  Print timestamp for debug output.
- void wpa_printf (int level, char *fmt,...) PRINTF_FORMAT(2
  conditional printf
- void void wpa_hexdump (int level, const char *title, const u8 *buf, size_t len)
  conditional hex dump
- void wpa_hexdump_key (int level, const char *title, const u8 *buf, size_t len)
  conditional hex dump, hide keys
- void wpa_hexdump_ascii (int level, const char *title, const u8 *buf, size_t len)
  conditional hex dump
- void wpa_hexdump_ascii_key (int level, const char *title, const u8 *buf, size_t len)
  conditional hex dump, hide keys
- void wpa_msg (void *ctx, int level, char *fmt,...) PRINTF_FORMAT(3

Conditional printf for default target and ctrl_iface monitors.

- void void wpa_msg_ctrl (void *ctx, int level, char *fmt,...) PRINTF_FORMAT(3

  Conditional printf for ctrl_iface monitors.

- void wpa_msg_register_cb (wpa_msg_cb_func func)

  Register callback function for wpa_msg() messages.

- void **hostapd_logger** (void *ctx, const u8 *addr, unsigned int module, int level, const char *fmt,...) PRINTF_FORMAT(5
- void hostapd_logger_register_cb (hostapd_logger_cb_func func)

  Register callback function for hostapd_logger().

Variables

- void void typedef void(* **wpa_msg_cb_func**)(void *ctx, int level, const char *txt, size_t len)
- void typedef void(* **hostapd_logger_cb_func**)(void *ctx, const u8 *addr, unsigned int module, int level, const char *txt, size_t len)

15.342.1 Detailed Description

wpa_supplicant/hostapd / Debug prints

Copyright

Copyright (c) 2002-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.342.2 Function Documentation

15.342.2.1 void hostapd_logger_register_cb (hostapd_logger_cb_func *func*)

Register callback function for hostapd_logger().

Parameters:

func Callback function (NULL to unregister)

15.342.2.2 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.

15.342.2.3 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.

15.342.2.4 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.

15.342.2.5 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.

15.342.2.6 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.

15.342.2.7 void wpa_msg (void * *ctx*, int *level*, char * *fmt*, ...)

Conditional printf for default target and ctrl_iface monitors.

Parameters:

ctx Pointer to context data; this is the ctx variable registered with struct wpa_driver_ops::init()
level priority level (MSG_*) of the message
fmt printf format string, followed by optional arguments

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration. This function is like wpa_printf(), but it also sends the same message to all attached ctrl_iface monitors.

Note: New line ’

’ is added to the end of the text when printing to stdout.

15.342.2.8 void void wpa_msg_ctrl (void * *ctx*, int *level*, char * *fmt*, ...)

Conditional printf for ctrl_iface monitors.

Parameters:

ctx Pointer to context data; this is the ctx variable registered with struct wpa_driver_ops::init()
level priority level (MSG_*) of the message
fmt printf format string, followed by optional arguments

This function is used to print conditional debugging and error messages. This function is like wpa_msg(), but it sends the output only to the attached ctrl_iface monitors. In other words, it can be used for frequent events that do not need to be sent to syslog.

15.342.2.9 void wpa_msg_register_cb (wpa_msg_cb_func *func*)

Register callback function for wpa_msg() messages.

Parameters:

func Callback function (NULL to unregister)

15.342.2.10 void wpa_printf (int *level*, char * *fmt*, ...)

conditional printf

Parameters:

level priority level (MSG_*) of the message

fmt printf format string, followed by optional arguments

This function is used to print conditional debugging and error messages. The output may be directed to stdout, stderr, and/or syslog based on configuration.

Note: New line ’

’ is added to the end of the text when printing to stdout.

15.343 src/utils/wpabuf.c File Reference

Dynamic data buffer. `#include "includes.h"`

`#include "common.h"`

`#include "wpabuf.h"`

Functions

- int **wpabuf_resize** (struct wpabuf **_buf, size_t add_len)
- struct wpabuf * wpabuf_alloc (size_t len)

  Allocate a wpabuf of the given size.

- struct wpabuf * **wpabuf_alloc_ext_data** (u8 *data, size_t len)
- struct wpabuf * **wpabuf_alloc_copy** (const void *data, size_t len)
- struct wpabuf * **wpabuf_dup** (const struct wpabuf *src)
- void wpabuf_free (struct wpabuf *buf)

  Free a wpabuf.

- void * **wpabuf_put** (struct wpabuf *buf, size_t len)
- struct wpabuf * wpabuf_concat (struct wpabuf *a, struct wpabuf *b)

  Concatenate two buffers into a newly allocated one.

- struct wpabuf * wpabuf_zeropad (struct wpabuf *buf, size_t len)

  Pad buffer with 0x00 octets (prefix) to specified length.

- void **wpabuf_printf** (struct wpabuf *buf, char *fmt,...)

15.343.1 Detailed Description

Dynamic data buffer.

Copyright

Copyright (c) 2007-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.343.2 Function Documentation

15.343.2.1 struct wpabuf* wpabuf_alloc (size_t *len*) [read]

Allocate a wpabuf of the given size.

Parameters:

len Length for the allocated buffer

Returns:

Buffer to the allocated wpabuf or NULL on failure

15.343.2.2 struct wpabuf* wpabuf_concat (struct wpabuf * *a*, struct wpabuf * *b*) [read]

Concatenate two buffers into a newly allocated one.

Parameters:

a First buffer

b Second buffer

Returns:

wpabuf with concatenated a + b data or NULL on failure

Both buffers a and b will be freed regardless of the return value. Input buffers can be NULL which is interpreted as an empty buffer.

15.343.2.3 void wpabuf_free (struct wpabuf * *buf*)

Free a wpabuf.

Parameters:

buf wpabuf buffer

15.343.2.4 struct wpabuf* wpabuf_zeropad (struct wpabuf * *buf*, size_t *len*) [read]

Pad buffer with 0x00 octets (prefix) to specified length.

Parameters:

buf Buffer to be padded

len Length for the padded buffer

Returns:

wpabuf padded to len octets or NULL on failure

If buf is longer than len octets or of same size, it will be returned as-is. Otherwise a new buffer is allocated and prefixed with 0x00 octets followed by the source data. The source buffer will be freed on error, i.e., caller will only be responsible on freeing the returned buffer. If buf is NULL, NULL will be returned.

15.344 src/utils/wpabuf.h File Reference

Dynamic data buffer.

Data Structures

- struct wpabuf

Functions

- int **wpabuf_resize** (struct wpabuf **buf, size_t add_len)
- struct wpabuf * wpabuf_alloc (size_t len)

  Allocate a wpabuf of the given size.

- struct wpabuf * **wpabuf_alloc_ext_data** (u8 *data, size_t len)
- struct wpabuf * **wpabuf_alloc_copy** (const void *data, size_t len)
- struct wpabuf * **wpabuf_dup** (const struct wpabuf *src)
- void wpabuf_free (struct wpabuf *buf)

  Free a wpabuf.

- void * **wpabuf_put** (struct wpabuf *buf, size_t len)
- struct wpabuf * wpabuf_concat (struct wpabuf *a, struct wpabuf *b)

  Concatenate two buffers into a newly allocated one.

- struct wpabuf * wpabuf_zeropad (struct wpabuf *buf, size_t len)

  Pad buffer with 0x00 octets (prefix) to specified length.

- void **wpabuf_printf** (struct wpabuf *buf, char *fmt,...) PRINTF_FORMAT(2

15.344.1 Detailed Description

Dynamic data buffer.

Copyright

Copyright (c) 2007-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.344.2 Function Documentation

15.344.2.1 struct wpabuf* wpabuf_alloc (size_t *len*) [read]

Allocate a wpabuf of the given size.

Parameters:

len Length for the allocated buffer

Returns:

Buffer to the allocated wpabuf or NULL on failure

15.344.2.2 struct wpabuf* wpabuf_concat (struct wpabuf * *a*, struct wpabuf * *b*) [read]

Concatenate two buffers into a newly allocated one.

Parameters:

a First buffer

b Second buffer

Returns:

wpabuf with concatenated a + b data or NULL on failure

Both buffers a and b will be freed regardless of the return value. Input buffers can be NULL which is interpreted as an empty buffer.

15.344.2.3 void wpabuf_free (struct wpabuf * *buf*)

Free a wpabuf.

Parameters:

buf wpabuf buffer

15.344.2.4 struct wpabuf* wpabuf_zeropad (struct wpabuf * *buf*, size_t *len*) [read]

Pad buffer with 0x00 octets (prefix) to specified length.

Parameters:

buf Buffer to be padded

len Length for the padded buffer

Returns:

wpabuf padded to len octets or NULL on failure

If buf is longer than len octets or of same size, it will be returned as-is. Otherwise a new buffer is allocated and prefixed with 0x00 octets followed by the source data. The source buffer will be freed on error, i.e., caller will only be responsible on freeing the returned buffer. If buf is NULL, NULL will be returned.

15.345 src/wps/http.h File Reference

HTTP for WPS.

Enumerations

- enum **http_reply_code** {

  HTTP_OK = 200, **HTTP_BAD_REQUEST** = 400, **UPNP_INVALID_ACTION** = 401, **UPNP_-INVALID_ARGS** = 402,

  HTTP_PRECONDITION_FAILED = 412, **HTTP_INTERNAL_SERVER_ERROR** = 500, **HTTP_UNIMPLEMENTED** = 501, **UPNP_ACTION_FAILED** = 501,

  UPNP_ARG_VALUE_INVALID = 600, **UPNP_ARG_VALUE_OUT_OF_RANGE** = 601, **UPNP_OUT_OF_MEMORY** = 603 }

15.345.1 Detailed Description

HTTP for WPS.

Copyright

Copyright (c) 2000-2003 Intel Corporation Copyright (c) 2006-2007 Sony Corporation Copyright (c) 2008-2009 Atheros Communications Copyright (c) 2009, Jouni Malinen <j@w1.fi>

See wps_upnp.c for more details on licensing and code history.

15.346 src/wps/http_client.c File Reference

http_client - HTTP client `#include "includes.h"`

```
#include <fcntl.h>
#include "common.h"
#include "eloop.h"
#include "httpread.h"
#include "http_client.h"
```

Data Structures

- struct http_client

Defines

- #define **HTTP_CLIENT_TIMEOUT** 30

Functions

- struct http_client * **http_client_addr** (struct sockaddr_in *dst, struct wpabuf *req, size_t max_response, void(*cb)(void *ctx, struct http_client *c, enum http_client_event event), void *cb_ctx)
- char * **http_client_url_parse** (const char *url, struct sockaddr_in *dst, char **ret_path)
- struct http_client * **http_client_url** (const char *url, struct wpabuf *req, size_t max_response, void(*cb)(void *ctx, struct http_client *c, enum http_client_event event), void *cb_ctx)
- void **http_client_free** (struct http_client *c)
- struct wpabuf * **http_client_get_body** (struct http_client *c)
- char * **http_link_update** (char *url, const char *base)

15.346.1 Detailed Description

http_client - HTTP client

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.347 src/wps/http_client.h File Reference

http_client - HTTP client

Enumerations

- enum **http_client_event** { **HTTP_CLIENT_FAILED**, **HTTP_CLIENT_TIMEOUT**, **HTTP_CLIENT_OK**, **HTTP_CLIENT_INVALID_REPLY** }

Functions

- char * **http_client_url_parse** (const char *url, struct sockaddr_in *dst, char **path)
- struct http_client * **http_client_addr** (struct sockaddr_in *dst, struct wpabuf *req, size_t max_response, void(*cb)(void *ctx, struct http_client *c, enum http_client_event event), void *cb_ctx)
- struct http_client * **http_client_url** (const char *url, struct wpabuf *req, size_t max_response, void(*cb)(void *ctx, struct http_client *c, enum http_client_event event), void *cb_ctx)
- void **http_client_free** (struct http_client *c)
- struct wpabuf * **http_client_get_body** (struct http_client *c)
- char * **http_link_update** (char *url, const char *base)

15.347.1 Detailed Description

http_client - HTTP client

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.348 src/wps/http_server.c File Reference

http_server - HTTP server `#include "includes.h"`

```
#include <fcntl.h>
#include "common.h"
#include "eloop.h"
#include "httpread.h"
#include "http_server.h"
```

Data Structures

- struct http_request
- struct http_server

Defines

- #define **HTTP_SERVER_TIMEOUT** 30
- #define **HTTP_SERVER_MAX_REQ_LEN** 8000
- #define **HTTP_SERVER_MAX_CONNECTIONS** 10

Functions

- void **http_request_deinit** (struct http_request *req)
- void **http_request_send** (struct http_request *req, struct wpabuf *resp)
- void **http_request_send_and_deinit** (struct http_request *req, struct wpabuf *resp)
- enum httpread_hdr_type **http_request_get_type** (struct http_request *req)
- char * **http_request_get_uri** (struct http_request *req)
- char * **http_request_get_hdr** (struct http_request *req)
- char * **http_request_get_data** (struct http_request *req)
- char * **http_request_get_hdr_line** (struct http_request *req, const char *tag)
- struct sockaddr_in * **http_request_get_cli_addr** (struct http_request *req)
- struct http_server * **http_server_init** (struct in_addr *addr, int port, void(*cb)(void *ctx, struct http_request *req), void *cb_ctx)
- void **http_server_deinit** (struct http_server *srv)
- int **http_server_get_port** (struct http_server *srv)

15.348.1 Detailed Description

http_server - HTTP server

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.349 src/wps/http_server.h File Reference

http_server - HTTP server

Functions

- void **http_request_deinit** (struct http_request *req)
- void **http_request_send** (struct http_request *req, struct wpabuf *resp)
- void **http_request_send_and_deinit** (struct http_request *req, struct wpabuf *resp)
- enum httpread_hdr_type **http_request_get_type** (struct http_request *req)
- char * **http_request_get_uri** (struct http_request *req)
- char * **http_request_get_hdr** (struct http_request *req)
- char * **http_request_get_data** (struct http_request *req)
- char * **http_request_get_hdr_line** (struct http_request *req, const char *tag)
- struct sockaddr_in * **http_request_get_cli_addr** (struct http_request *req)
- struct http_server * **http_server_init** (struct in_addr *addr, int port, void(*cb)(void *ctx, struct http_request *req), void *cb_ctx)
- void **http_server_deinit** (struct http_server *srv)
- int **http_server_get_port** (struct http_server *srv)

15.349.1 Detailed Description

http_server - HTTP server

Copyright

15.350 src/wps/httpread.c File Reference

httpread - Manage reading file(s) from HTTP/TCP socket Author: Ted Merrill `#include "includes.h"`

`#include "common.h"`

`#include "eloop.h"`

`#include "httpread.h"`

Data Structures

- struct httpread

Defines

- #define **HTTPREAD_READBUF_SIZE** 1024
- #define **HTTPREAD_HEADER_MAX_SIZE** 4096
- #define **HTTPREAD_BODYBUF_DELTA** 4096
- #define **httpread_debug** 0

Functions

- void **httpread_destroy** (struct httpread *h)
- struct httpread * **httpread_create** (int sd, void(*cb)(struct httpread *handle, void *cookie, enum httpread_event e), void *cookie, int max_bytes, int timeout_seconds)
- enum httpread_hdr_type **httpread_hdr_type_get** (struct httpread *h)
- char * **httpread_uri_get** (struct httpread *h)
- int **httpread_reply_code_get** (struct httpread *h)
- int **httpread_length_get** (struct httpread *h)
- void * **httpread_data_get** (struct httpread *h)
- char * **httpread_hdr_get** (struct httpread *h)
- char * **httpread_hdr_line_get** (struct httpread *h, const char *tag)

15.350.1 Detailed Description

httpread - Manage reading file(s) from HTTP/TCP socket Author: Ted Merrill

Copyright

The files are buffered via internal callbacks from eloop, then presented to an application callback routine when completely read into memory. May also be used if no file is expected but just to get the header, including HTTP replies (e.g. HTTP/1.1 200 OK etc.).

This does not attempt to be an optimally efficient implementation, but does attempt to be of reasonably small size and memory consumption; assuming that only small files are to be read. A maximum file size is provided by application and enforced.

It is assumed that the application does not expect any of the following: -- transfer encoding other than chunked -- trailer fields It is assumed that, even if the other side requested that the connection be kept open, that we will close it (thus HTTP messages sent by application should have the connection closed field); this is allowed by HTTP/1.1 and simplifies things for us.

Other limitations: -- HTTP header may not exceed a hard-coded size.

Notes: This code would be massively simpler without some of the new features of HTTP/1.1, especially chunked data.

15.351 src/wps/httpread.h File Reference

httpread - Manage reading file(s) from HTTP/TCP socket Author: Ted Merrill

Enumerations

- enum **httpread_event** { **HTTPREAD_EVENT_FILE_READY** = 1, **HTTPREAD_EVENT_-TIMEOUT** = 2, **HTTPREAD_EVENT_ERROR** = 3 }
- enum **httpread_hdr_type** {

  HTTPREAD_HDR_TYPE_UNKNOWN = 0, **HTTPREAD_HDR_TYPE_REPLY** = 1, **HTTPREAD_HDR_TYPE_GET** = 2, **HTTPREAD_HDR_TYPE_HEAD** = 3,

  HTTPREAD_HDR_TYPE_POST = 4, **HTTPREAD_HDR_TYPE_PUT** = 5, **HTTPREAD_-HDR_TYPE_DELETE** = 6, **HTTPREAD_HDR_TYPE_TRACE** = 7,

  HTTPREAD_HDR_TYPE_CONNECT = 8, **HTTPREAD_HDR_TYPE_NOTIFY** = 9, **HTTPREAD_HDR_TYPE_M_SEARCH** = 10, **HTTPREAD_HDR_TYPE_M_POST** = 11,

  HTTPREAD_HDR_TYPE_SUBSCRIBE = 12, **HTTPREAD_HDR_TYPE_UNSUBSCRIBE** = 13, **HTTPREAD_N_HDR_TYPES** }

Functions

- void **httpread_destroy** (struct httpread *h)
- struct httpread * **httpread_create** (int sd, void(*cb)(struct httpread *handle, void *cookie, enum httpread_event e), void *cookie, int max_bytes, int timeout_seconds)
- enum httpread_hdr_type **httpread_hdr_type_get** (struct httpread *h)
- char * **httpread_uri_get** (struct httpread *h)
- int **httpread_reply_code_get** (struct httpread *h)
- int **httpread_length_get** (struct httpread *h)
- void * **httpread_data_get** (struct httpread *h)
- char * **httpread_hdr_get** (struct httpread *h)
- char * **httpread_hdr_line_get** (struct httpread *h, const char *tag)

15.351.1 Detailed Description

httpread - Manage reading file(s) from HTTP/TCP socket Author: Ted Merrill

Copyright

15.352 src/wps/ndef.c File Reference

NDEF(NFC Data Exchange Format) routines for Wi-Fi Protected Setup Reference is "NFCForum-TS-NDEF_1.0 2006-07-24". `#include "includes.h"`

```
#include "common.h"
#include "wps/wps.h"
#include "wps/wps_i.h"
```

Data Structures

- struct ndef_record

Defines

- #define **FLAG_MESSAGE_BEGIN** (1 << 7)
- #define **FLAG_MESSAGE_END** (1 << 6)
- #define **FLAG_CHUNK** (1 << 5)
- #define **FLAG_SHORT_RECORD** (1 << 4)
- #define **FLAG_ID_LENGTH_PRESENT** (1 << 3)
- #define **FLAG_TNF_RFC2046** (0x02)

Functions

- struct wpabuf * **ndef_parse_wifi** (struct wpabuf *buf)
- struct wpabuf * **ndef_build_wifi** (struct wpabuf *buf)

15.352.1 Detailed Description

NDEF(NFC Data Exchange Format) routines for Wi-Fi Protected Setup Reference is "NFCForum-TS-NDEF_1.0 2006-07-24".

Copyright

15.353 src/wps/upnp_xml.c File Reference

UPnP XML helper routines. `#include "includes.h"`

`#include "common.h"`

`#include "base64.h"`

`#include "http.h"`

`#include "upnp_xml.h"`

Functions

- void **xml_data_encode** (struct wpabuf *buf, const char *data, int len)
- void **xml_add_tagged_data** (struct wpabuf *buf, const char *tag, const char *data)
- char * **xml_get_first_item** (const char *doc, const char *item)
- struct wpabuf * **xml_get_base64_item** (const char *data, const char *name, enum http_reply_code *ret)

15.353.1 Detailed Description

UPnP XML helper routines.

Copyright

Copyright (c) 2000-2003 Intel Corporation Copyright (c) 2006-2007 Sony Corporation Copyright (c) 2008-2009 Atheros Communications Copyright (c) 2009, Jouni Malinen <j@w1.fi>

See wps_upnp.c for more details on licensing and code history.

15.354 src/wps/upnp_xml.h File Reference

UPnP XML helper routines. `#include "http.h"`

Functions

- void **xml_data_encode** (struct wpabuf *buf, const char *data, int len)
- void **xml_add_tagged_data** (struct wpabuf *buf, const char *tag, const char *data)
- char * **xml_get_first_item** (const char *doc, const char *item)
- struct wpabuf * **xml_get_base64_item** (const char *data, const char *name, enum http_reply_code *ret)

15.354.1 Detailed Description

UPnP XML helper routines.

Copyright

Copyright (c) 2000-2003 Intel Corporation Copyright (c) 2006-2007 Sony Corporation Copyright (c) 2008-2009 Atheros Communications Copyright (c) 2009, Jouni Malinen <j@w1.fi>

See wps_upnp.c for more details on licensing and code history.

15.355 src/wps/wps.c File Reference

Wi-Fi Protected Setup. `#include "includes.h"`

`#include "common.h"`

`#include "dh_group5.h"`

`#include "wps_i.h"`

`#include "wps_dev_attr.h"`

`#include "ieee802_11_defs.h"`

Functions

- struct wps_data * wps_init (const struct wps_config *cfg)

  Initialize WPS Registration protocol data.

- void wps_deinit (struct wps_data *data)

  Deinitialize WPS Registration protocol data.

- enum wps_process_res wps_process_msg (struct wps_data *wps, enum wsc_op_code op_code, const struct wpabuf *msg)

  Process a WPS message.

- struct wpabuf * wps_get_msg (struct wps_data *wps, enum wsc_op_code *op_code)

  Build a WPS message.

- int wps_is_selected_pbc_registrar (const struct wpabuf *msg)

  Check whether WPS IE indicates active PBC.

- int wps_is_selected_pin_registrar (const struct wpabuf *msg)

  Check whether WPS IE indicates active PIN.

- const u8 * wps_get_uuid_e (const struct wpabuf *msg)

  Get UUID-E from WPS IE.

- struct wpabuf * wps_build_assoc_req_ie (enum wps_request_type req_type)

  Build WPS IE for (Re)Association Request.

- struct wpabuf * wps_build_probe_req_ie (int pbc, struct wps_device_data *dev, const u8 *uuid, enum wps_request_type req_type)

  Build WPS IE for Probe Request.

- void **wps_free_pending_msgs** (struct upnp_pending_message *msgs)
- int **wps_attr_text** (struct wpabuf *data, char *buf, char *end)

15.355.1 Detailed Description

Wi-Fi Protected Setup.

Copyright

15.355.2 Function Documentation

15.355.2.1 struct wpabuf* wps_build_assoc_req_ie (enum wps_request_type *req_type*) [read]

Build WPS IE for (Re)Association Request.

Parameters:

req_type Value for Request Type attribute

Returns:

WPS IE or NULL on failure

The caller is responsible for freeing the buffer.

15.355.2.2 struct wpabuf* wps_build_probe_req_ie (int *pbc*, struct wps_device_data * *dev*, const u8 * *uuid*, enum wps_request_type *req_type*) [read]

Build WPS IE for Probe Request.

Parameters:

pbc Whether searching for PBC mode APs

dev Device attributes

uuid Own UUID

req_type Value for Request Type attribute

Returns:

WPS IE or NULL on failure

The caller is responsible for freeing the buffer.

15.355.2.3 void wps_deinit (struct wps_data * *data*)

Deinitialize WPS Registration protocol data.

Parameters:

data WPS Registration protocol data from wps_init()

15.355.2.4 struct wpabuf* wps_get_msg (struct wps_data * *wps*, enum wsc_op_code * *op_code*) [read]

Build a WPS message.

Parameters:

wps WPS Registration protocol data from wps_init()

op_code Buffer for returning message OP Code

Returns:

The generated WPS message or NULL on failure

This function is used to build a response to a message processed by calling wps_process_msg(). The caller is responsible for freeing the buffer.

15.355.2.5 const u8* wps_get_uuid_e (const struct wpabuf * *msg*)

Get UUID-E from WPS IE.

Parameters:

msg WPS IE contents from Beacon or Probe Response frame

Returns:

Pointer to UUID-E or NULL if not included

The returned pointer is to the msg contents and it remains valid only as long as the msg buffer is valid.

15.355.2.6 struct wps_data* wps_init (const struct wps_config * *cfg*) [read]

Initialize WPS Registration protocol data.

Parameters:

cfg WPS configuration

Returns:

Pointer to allocated data or NULL on failure

This function is used to initialize WPS data for a registration protocol instance (i.e., each run of registration protocol as a Registrar of Enrollee. The caller is responsible for freeing this data after the registration run has been completed by calling wps_deinit().

15.355.2.7 int wps_is_selected_pbc_registrar (const struct wpabuf * *msg*)

Check whether WPS IE indicates active PBC.

Parameters:

msg WPS IE contents from Beacon or Probe Response frame

Returns:

1 if PBC Registrar is active, 0 if not

15.355.2.8 int wps_is_selected_pin_registrar (const struct wpabuf * *msg*)

Check whether WPS IE indicates active PIN.

Parameters:

msg WPS IE contents from Beacon or Probe Response frame

Returns:

1 if PIN Registrar is active, 0 if not

15.355.2.9 enum wps_process_res wps_process_msg (struct wps_data * *wps*, enum wsc_op_code *op_code*, const struct wpabuf * *msg*)

Process a WPS message.

Parameters:

wps WPS Registration protocol data from wps_init()

op_code Message OP Code

msg Message data

Returns:

Processing result

This function is used to process WPS messages with OP Codes WSC_ACK, WSC_NACK, WSC_MSG, and WSC_Done. The caller (e.g., EAP server/peer) is responsible for reassembling the messages before calling this function. Response to this message is built by calling wps_get_msg().

15.356 src/wps/wps.h File Reference

Wi-Fi Protected Setup. `#include "wps_defs.h"`

Data Structures

- struct wps_credential

  WPS Credential.

- struct wps_device_data

  WPS Device Data.

- struct oob_conf_data
- struct wps_config

  WPS configuration for a single registration protocol run.

- struct wps_registrar_config

  WPS Registrar configuration.

- union wps_event_data
- struct wps_event_data::wps_event_m2d

  M2D event data.

- struct wps_event_data::wps_event_fail

  Registration failure information.

- struct wps_event_data::wps_event_pwd_auth_fail
- struct wps_event_data::wps_event_er_ap
- struct wps_event_data::wps_event_er_enrollee
- struct upnp_pending_message

  Pending PutWLANResponse messages.

- struct wps_context

  Long term WPS context data.

- struct oob_device_data
- struct oob_nfc_device_data

Defines

- #define **WPS_DEV_TYPE_LEN** 8
- #define **WPS_DEV_TYPE_BUFSIZE** 21

Enumerations

- enum wsc_op_code {

  WSC_UPnP = 0, **WSC_Start** = 0x01, **WSC_ACK** = 0x02, **WSC_NACK** = 0x03,

  WSC_MSG = 0x04, **WSC_Done** = 0x05, **WSC_FRAG_ACK** = 0x06 }

- enum wps_process_res { WPS_DONE, WPS_CONTINUE, WPS_FAILURE, WPS_PENDING }
- enum wps_event {
  WPS_EV_M2D, WPS_EV_FAIL, WPS_EV_SUCCESS, WPS_EV_PWD_AUTH_FAIL,
  WPS_EV_PBC_OVERLAP, WPS_EV_PBC_TIMEOUT, WPS_EV_ER_AP_ADD, WPS_EV_-ER_AP_REMOVE,
  WPS_EV_ER_ENROLLEE_ADD, WPS_EV_ER_ENROLLEE_REMOVE }

Functions

- struct wps_data * wps_init (const struct wps_config *cfg)
  Initialize WPS Registration protocol data.
- void wps_deinit (struct wps_data *data)
  Deinitialize WPS Registration protocol data.
- enum wps_process_res wps_process_msg (struct wps_data *wps, enum wsc_op_code op_code, const struct wpabuf *msg)
  Process a WPS message.
- struct wpabuf * wps_get_msg (struct wps_data *wps, enum wsc_op_code *op_code)
  Build a WPS message.
- int wps_is_selected_pbc_registrar (const struct wpabuf *msg)
  Check whether WPS IE indicates active PBC.
- int wps_is_selected_pin_registrar (const struct wpabuf *msg)
  Check whether WPS IE indicates active PIN.
- const u8 * wps_get_uuid_e (const struct wpabuf *msg)
  Get UUID-E from WPS IE.
- struct wpabuf * wps_build_assoc_req_ie (enum wps_request_type req_type)
  Build WPS IE for (Re)Association Request.
- struct wpabuf * wps_build_probe_req_ie (int pbc, struct wps_device_data *dev, const u8 *uuid, enum wps_request_type req_type)
  Build WPS IE for Probe Request.
- struct wps_registrar * wps_registrar_init (struct wps_context *wps, const struct wps_registrar_config *cfg)
  Initialize WPS Registrar data.
- void wps_registrar_deinit (struct wps_registrar *reg)
  Deinitialize WPS Registrar data.
- int wps_registrar_add_pin (struct wps_registrar *reg, const u8 *uuid, const u8 *pin, size_t pin_len, int timeout)
  Configure a new PIN for Registrar.

- int wps_registrar_invalidate_pin (struct wps_registrar *reg, const u8 *uuid)
  Invalidate a PIN for a specific UUID-E.
- int wps_registrar_unlock_pin (struct wps_registrar *reg, const u8 *uuid)
  Unlock a PIN for a specific UUID-E.
- int wps_registrar_button_pushed (struct wps_registrar *reg)
  Notify Registrar that AP button was pushed.
- void wps_registrar_probe_req_rx (struct wps_registrar *reg, const u8 *addr, const struct wpabuf *wps_data)
  Notify Registrar of Probe Request.
- int **wps_registrar_update_ie** (struct wps_registrar *reg)
- int wps_registrar_set_selected_registrar (struct wps_registrar *reg, const struct wpabuf *msg)
  Notification of SetSelectedRegistrar.
- int **wps_registrar_get_info** (struct wps_registrar *reg, const u8 *addr, char *buf, size_t buflen)
- unsigned int wps_pin_checksum (unsigned int pin)
  Compute PIN checksum.
- unsigned int wps_pin_valid (unsigned int pin)
  Check whether a PIN has a valid checksum.
- unsigned int wps_generate_pin (void)
  Generate a random PIN.
- void **wps_free_pending_msgs** (struct upnp_pending_message *msgs)
- struct oob_device_data * **wps_get_oob_device** (char *device_type)
- struct oob_nfc_device_data * **wps_get_oob_nfc_device** (char *device_name)
- int **wps_get_oob_method** (char *method)
- int **wps_process_oob** (struct wps_context *wps, struct oob_device_data *oob_dev, int registrar)
- int **wps_attr_text** (struct wpabuf *data, char *buf, char *end)
- struct wps_er * **wps_er_init** (struct wps_context *wps, const char *ifname)
- void **wps_er_refresh** (struct wps_er *er)
- void **wps_er_deinit** (struct wps_er *er)
- void **wps_er_set_sel_reg** (struct wps_er *er, int sel_reg, u16 dev_passwd_id, u16 sel_reg_config_methods)
- int **wps_er_pbc** (struct wps_er *er, const u8 *uuid)
- int **wps_er_learn** (struct wps_er *er, const u8 *uuid, const u8 *pin, size_t pin_len)
- int **wps_dev_type_str2bin** (const char *str, u8 dev_type[WPS_DEV_TYPE_LEN])
- char * **wps_dev_type_bin2str** (const u8 dev_type[WPS_DEV_TYPE_LEN], char *buf, size_t buf_len)

15.356.1 Detailed Description

Wi-Fi Protected Setup.

Copyright

Copyright (c) 2007-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.356.2 Enumeration Type Documentation

15.356.2.1 enum wps_event

enum wps_event - WPS event types

Enumerator:

WPS_EV_M2D M2D received (Registrar did not know us).

WPS_EV_FAIL Registration failed.

WPS_EV_SUCCESS Registration succeeded.

WPS_EV_PWD_AUTH_FAIL Password authentication failed.

WPS_EV_PBC_OVERLAP PBC session overlap detected.

WPS_EV_PBC_TIMEOUT PBC walktime expired before protocol run start.

WPS_EV_ER_AP_ADD ER: AP added.

WPS_EV_ER_AP_REMOVE ER: AP removed.

WPS_EV_ER_ENROLLEE_ADD ER: Enrollee added.

WPS_EV_ER_ENROLLEE_REMOVE ER: Enrollee removed.

15.356.2.2 enum wps_process_res

enum wps_process_res - WPS message processing result

Enumerator:

WPS_DONE Processing done.

WPS_CONTINUE Processing continues.

WPS_FAILURE Processing failed.

WPS_PENDING Processing continues, but waiting for an external. event (e.g., UPnP message from an external Registrar)

15.356.2.3 enum wsc_op_code

enum wsc_op_code - EAP-WSC OP-Code values

15.356.3 Function Documentation

15.356.3.1 struct wpabuf* wps_build_assoc_req_ie (enum wps_request_type *req_type*) [read]

Build WPS IE for (Re)Association Request.

Parameters:

req_type Value for Request Type attribute

Returns:

WPS IE or NULL on failure

The caller is responsible for freeing the buffer.

15.356.3.2 struct wpabuf* wps_build_probe_req_ie (int *pbc*, struct wps_device_data * *dev*, const u8 * *uuid*, enum wps_request_type *req_type*) [read]

Build WPS IE for Probe Request.

Parameters:

pbc Whether searching for PBC mode APs

dev Device attributes

uuid Own UUID

req_type Value for Request Type attribute

Returns:

WPS IE or NULL on failure

The caller is responsible for freeing the buffer.

15.356.3.3 void wps_deinit (struct wps_data * *data*)

Deinitialize WPS Registration protocol data.

Parameters:

data WPS Registration protocol data from wps_init()

15.356.3.4 unsigned int wps_generate_pin (void)

Generate a random PIN.

Returns:

Eight digit PIN (i.e., including the checksum digit)

15.356.3.5 struct wpabuf* wps_get_msg (struct wps_data * *wps*, enum wsc_op_code * *op_code*) [read]

Build a WPS message.

Parameters:

wps WPS Registration protocol data from wps_init()

op_code Buffer for returning message OP Code

Returns:

The generated WPS message or NULL on failure

This function is used to build a response to a message processed by calling wps_process_msg(). The caller is responsible for freeing the buffer.

15.356.3.6 const u8* wps_get_uuid_e (const struct wpabuf * *msg*)

Get UUID-E from WPS IE.

Parameters:

msg WPS IE contents from Beacon or Probe Response frame

Returns:

Pointer to UUID-E or NULL if not included

The returned pointer is to the msg contents and it remains valid only as long as the msg buffer is valid.

15.356.3.7 struct wps_data* wps_init (const struct wps_config * *cfg*) [read]

Initialize WPS Registration protocol data.

Parameters:

cfg WPS configuration

Returns:

Pointer to allocated data or NULL on failure

This function is used to initialize WPS data for a registration protocol instance (i.e., each run of registration protocol as a Registrar of Enrollee. The caller is responsible for freeing this data after the registration run has been completed by calling wps_deinit().

15.356.3.8 int wps_is_selected_pbc_registrar (const struct wpabuf * *msg*)

Check whether WPS IE indicates active PBC.

Parameters:

msg WPS IE contents from Beacon or Probe Response frame

Returns:

1 if PBC Registrar is active, 0 if not

15.356.3.9 int wps_is_selected_pin_registrar (const struct wpabuf * *msg*)

Check whether WPS IE indicates active PIN.

Parameters:

msg WPS IE contents from Beacon or Probe Response frame

Returns:

1 if PIN Registrar is active, 0 if not

15.356.3.10 unsigned int wps_pin_checksum (unsigned int *pin*)

Compute PIN checksum.

Parameters:

pin Seven digit PIN (i.e., eight digit PIN without the checksum digit)

Returns:

Checksum digit

15.356.3.11 unsigned int wps_pin_valid (unsigned int *pin*)

Check whether a PIN has a valid checksum.

Parameters:

pin Eight digit PIN (i.e., including the checksum digit)

Returns:

1 if checksum digit is valid, or 0 if not

15.356.3.12 enum wps_process_res wps_process_msg (struct wps_data * *wps*, enum wsc_op_code *op_code*, const struct wpabuf * *msg*)

Process a WPS message.

Parameters:

wps WPS Registration protocol data from wps_init()

op_code Message OP Code

msg Message data

Returns:

Processing result

This function is used to process WPS messages with OP Codes WSC_ACK, WSC_NACK, WSC_MSG, and WSC_Done. The caller (e.g., EAP server/peer) is responsible for reassembling the messages before calling this function. Response to this message is built by calling wps_get_msg().

15.356.3.13 int wps_registrar_add_pin (struct wps_registrar * *reg*, const u8 * *uuid*, const u8 * *pin*, size_t *pin_len*, int *timeout*)

Configure a new PIN for Registrar.

Parameters:

reg Registrar data from wps_registrar_init()

uuid UUID-E or NULL for wildcard (any UUID)

pin PIN (Device Password)

pin_len Length of pin in octets

timeout Time (in seconds) when the PIN will be invalidated; 0 = no timeout

Returns:

0 on success, -1 on failure

15.356.3.14 int wps_registrar_button_pushed (struct wps_registrar * *reg*)

Notify Registrar that AP button was pushed.

Parameters:

reg Registrar data from wps_registrar_init()

Returns:

0 on success, -1 on failure

This function is called on an AP when a push button is pushed to activate PBC mode. The PBC mode will be stopped after walk time (2 minutes) timeout or when a PBC registration is completed.

15.356.3.15 void wps_registrar_deinit (struct wps_registrar * *reg*)

Deinitialize WPS Registrar data.

Parameters:

reg Registrar data from wps_registrar_init()

15.356.3.16 struct wps_registrar* wps_registrar_init (struct wps_context * *wps*, const struct wps_registrar_config * *cfg*) [read]

Initialize WPS Registrar data.

Parameters:

wps Pointer to longterm WPS context

cfg Registrar configuration

Returns:

Pointer to allocated Registrar data or NULL on failure

This function is used to initialize WPS Registrar functionality. It can be used for a single Registrar run (e.g., when run in a supplicant) or multiple runs (e.g., when run as an internal Registrar in an AP). Caller is responsible for freeing the returned data with wps_registrar_deinit() when Registrar functionality is not needed anymore.

15.356.3.17 int wps_registrar_invalidate_pin (struct wps_registrar * *reg*, const u8 * *uuid*)

Invalidate a PIN for a specific UUID-E.

Parameters:

reg Registrar data from wps_registrar_init()

uuid UUID-E

Returns:

0 on success, -1 on failure (e.g., PIN not found)

15.356.3.18 void wps_registrar_probe_req_rx (struct wps_registrar * *reg*, const u8 * *addr*, const struct wpabuf * *wps_data*)

Notify Registrar of Probe Request.

Parameters:

reg Registrar data from wps_registrar_init()

addr MAC address of the Probe Request sender

wps_data WPS IE contents

This function is called on an AP when a Probe Request with WPS IE is received. This is used to track PBC mode use and to detect possible overlap situation with other WPS APs.

15.356.3.19 int wps_registrar_set_selected_registrar (struct wps_registrar * *reg*, const struct wpabuf * *msg*)

Notification of SetSelectedRegistrar.

Parameters:

reg Registrar data from wps_registrar_init()

msg Received message from SetSelectedRegistrar

Returns:

0 on success, -1 on failure

This function is called when an AP receives a SetSelectedRegistrar UPnP message.

15.356.3.20 int wps_registrar_unlock_pin (struct wps_registrar * *reg*, const u8 * *uuid*)

Unlock a PIN for a specific UUID-E.

Parameters:

reg Registrar data from wps_registrar_init()

uuid UUID-E

Returns:

0 on success, -1 on failure

PINs are locked to enforce only one concurrent use. This function unlocks a PIN to allow it to be used again. If the specified PIN was configured using a wildcard UUID, it will be removed instead of allowing multiple uses.

15.357 src/wps/wps_attr_build.c File Reference

Wi-Fi Protected Setup - attribute building. `#include "includes.h"`

```
#include "common.h"
#include "dh_group5.h"
#include "crypto.h"
#include "sha256.h"
#include "aes_wrap.h"
#include "wps_i.h"
```

Functions

- int **wps_build_public_key** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_req_type** (struct wpabuf *msg, enum wps_request_type type)
- int **wps_build_config_methods** (struct wpabuf *msg, u16 methods)
- int **wps_build_uuid_e** (struct wpabuf *msg, const u8 *uuid)
- int **wps_build_dev_password_id** (struct wpabuf *msg, u16 id)
- int **wps_build_config_error** (struct wpabuf *msg, u16 err)
- int **wps_build_authenticator** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_version** (struct wpabuf *msg)
- int **wps_build_msg_type** (struct wpabuf *msg, enum wps_msg_type msg_type)
- int **wps_build_enrollee_nonce** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_registrar_nonce** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_auth_type_flags** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_encr_type_flags** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_conn_type_flags** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_assoc_state** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_key_wrap_auth** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_encr_settings** (struct wps_data *wps, struct wpabuf *msg, struct wpabuf *plain)

15.357.1 Detailed Description

Wi-Fi Protected Setup - attribute building.

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.358 src/wps/wps_attr_parse.c File Reference

Wi-Fi Protected Setup - attribute parsing. `#include "includes.h"`

`#include "common.h"`

`#include "wps_i.h"`

Functions

- int **wps_parse_msg** (const struct wpabuf *msg, struct wps_parse_attr *attr)

15.358.1 Detailed Description

Wi-Fi Protected Setup - attribute parsing.

Copyright

15.359 src/wps/wps_attr_process.c File Reference

Wi-Fi Protected Setup - attribute processing. `#include "includes.h"`

`#include "common.h"`

`#include "sha256.h"`

`#include "wps_i.h"`

Functions

- int **wps_process_authenticator** (struct wps_data *wps, const u8 *authenticator, const struct wpabuf *msg)
- int **wps_process_key_wrap_auth** (struct wps_data *wps, struct wpabuf *msg, const u8 *key_wrap_auth)
- int **wps_process_cred** (struct wps_parse_attr *attr, struct wps_credential *cred)
- int **wps_process_ap_settings** (struct wps_parse_attr *attr, struct wps_credential *cred)

15.359.1 Detailed Description

Wi-Fi Protected Setup - attribute processing.

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.360 src/wps/wps_common.c File Reference

Wi-Fi Protected Setup - common functionality. `#include "includes.h"`

```
#include "common.h"
#include "dh_group5.h"
#include "sha256.h"
#include "aes_wrap.h"
#include "crypto.h"
#include "wps_i.h"
#include "wps_dev_attr.h"
```

Functions

- void **wps_kdf** (const u8 *key, const u8 *label_prefix, size_t label_prefix_len, const char *label, u8 *res, size_t res_len)
- int **wps_derive_keys** (struct wps_data *wps)
- void **wps_derive_psk** (struct wps_data *wps, const u8 *dev_passwd, size_t dev_passwd_len)
- struct wpabuf * **wps_decrypt_encr_settings** (struct wps_data *wps, const u8 *encr, size_t encr_len)

- unsigned int wps_pin_checksum (unsigned int pin)

  Compute PIN checksum.

- unsigned int wps_pin_valid (unsigned int pin)

  Check whether a PIN has a valid checksum.

- unsigned int wps_generate_pin (void)

  Generate a random PIN.

- void **wps_fail_event** (struct wps_context *wps, enum wps_msg_type msg)
- void **wps_success_event** (struct wps_context *wps)
- void **wps_pwd_auth_fail_event** (struct wps_context *wps, int enrollee, int part)
- void **wps_pbc_overlap_event** (struct wps_context *wps)
- void **wps_pbc_timeout_event** (struct wps_context *wps)
- int **wps_dev_type_str2bin** (const char *str, u8 dev_type[WPS_DEV_TYPE_LEN])
- char * **wps_dev_type_bin2str** (const u8 dev_type[WPS_DEV_TYPE_LEN], char *buf, size_t buf_len)

15.360.1 Detailed Description

Wi-Fi Protected Setup - common functionality.

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.360.2 Function Documentation

15.360.2.1 unsigned int wps_generate_pin (void)

Generate a random PIN.

Returns:

Eight digit PIN (i.e., including the checksum digit)

15.360.2.2 unsigned int wps_pin_checksum (unsigned int *pin*)

Compute PIN checksum.

Parameters:

pin Seven digit PIN (i.e., eight digit PIN without the checksum digit)

Returns:

Checksum digit

15.360.2.3 unsigned int wps_pin_valid (unsigned int *pin*)

Check whether a PIN has a valid checksum.

Parameters:

pin Eight digit PIN (i.e., including the checksum digit)

Returns:

1 if checksum digit is valid, or 0 if not

15.361 src/wps/wps_defs.h File Reference

Wi-Fi Protected Setup - message definitions.

Defines

- #define **WPS_VERSION** 0x10
- #define **WPS_DH_GROUP** 5
- #define **WPS_UUID_LEN** 16
- #define **WPS_NONCE_LEN** 16
- #define **WPS_AUTHENTICATOR_LEN** 8
- #define **WPS_AUTHKEY_LEN** 32
- #define **WPS_KEYWRAPKEY_LEN** 16
- #define **WPS_EMSK_LEN** 32
- #define **WPS_PSK_LEN** 16
- #define **WPS_SECRET_NONCE_LEN** 16
- #define **WPS_HASH_LEN** 32
- #define **WPS_KWA_LEN** 8
- #define **WPS_MGMTAUTHKEY_LEN** 32
- #define **WPS_MGMTENCKEY_LEN** 16
- #define **WPS_MGMT_KEY_ID_LEN** 16
- #define **WPS_OOB_DEVICE_PASSWORD_ATTR_LEN** 54
- #define **WPS_OOB_DEVICE_PASSWORD_LEN** 32
- #define **WPS_OOB_PUBKEY_HASH_LEN** 20
- #define **WPS_AUTH_OPEN** 0x0001
- #define **WPS_AUTH_WPAPSK** 0x0002
- #define **WPS_AUTH_SHARED** 0x0004
- #define **WPS_AUTH_WPA** 0x0008
- #define **WPS_AUTH_WPA2** 0x0010
- #define **WPS_AUTH_WPA2PSK** 0x0020
- #define **WPS_AUTH_TYPES**
- #define **WPS_ENCR_NONE** 0x0001
- #define **WPS_ENCR_WEP** 0x0002
- #define **WPS_ENCR_TKIP** 0x0004
- #define **WPS_ENCR_AES** 0x0008
- #define **WPS_ENCR_TYPES**
- #define **WPS_RF_24GHZ** 0x01
- #define **WPS_RF_50GHZ** 0x02
- #define **WPS_CONFIG_USBA** 0x0001
- #define **WPS_CONFIG_ETHERNET** 0x0002
- #define **WPS_CONFIG_LABEL** 0x0004
- #define **WPS_CONFIG_DISPLAY** 0x0008
- #define **WPS_CONFIG_EXT_NFC_TOKEN** 0x0010
- #define **WPS_CONFIG_INT_NFC_TOKEN** 0x0020
- #define **WPS_CONFIG_NFC_INTERFACE** 0x0040
- #define **WPS_CONFIG_PUSHBUTTON** 0x0080
- #define **WPS_CONFIG_KEYPAD** 0x0100
- #define **WPS_CONN_ESS** 0x01
- #define **WPS_CONN_IBSS** 0x02
- #define **WPS_DEV_OUI_WFA** 0x0050f204
- #define **WPS_PBC_WALK_TIME** 120

Enumerations

- enum **wps_attribute** {

  ATTR_AP_CHANNEL = 0x1001, **ATTR_ASSOC_STATE** = 0x1002, **ATTR_AUTH_TYPE** = 0x1003, **ATTR_AUTH_TYPE_FLAGS** = 0x1004,

  ATTR_AUTHENTICATOR = 0x1005, **ATTR_CONFIG_METHODS** = 0x1008, **ATTR_CONFIG_ERROR** = 0x1009, **ATTR_CONFIRM_URL4** = 0x100a,

  ATTR_CONFIRM_URL6 = 0x100b, **ATTR_CONN_TYPE** = 0x100c, **ATTR_CONN_TYPE_FLAGS** = 0x100d, **ATTR_CRED** = 0x100e,

  ATTR_ENCR_TYPE = 0x100f, **ATTR_ENCR_TYPE_FLAGS** = 0x1010, **ATTR_DEV_NAME** = 0x1011, **ATTR_DEV_PASSWORD_ID** = 0x1012,

  ATTR_E_HASH1 = 0x1014, **ATTR_E_HASH2** = 0x1015, **ATTR_E_SNONCE1** = 0x1016, **ATTR_E_SNONCE2** = 0x1017,

  ATTR_ENCR_SETTINGS = 0x1018, **ATTR_ENROLLEE_NONCE** = 0x101a, **ATTR_FEATURE_ID** = 0x101b, **ATTR_IDENTITY** = 0x101c,

  ATTR_IDENTITY_PROOF = 0x101d, **ATTR_KEY_WRAP_AUTH** = 0x101e, **ATTR_KEY_ID** = 0x101f, **ATTR_MAC_ADDR** = 0x1020,

  ATTR_MANUFACTURER = 0x1021, **ATTR_MSG_TYPE** = 0x1022, **ATTR_MODEL_NAME** = 0x1023, **ATTR_MODEL_NUMBER** = 0x1024,

  ATTR_NETWORK_INDEX = 0x1026, **ATTR_NETWORK_KEY** = 0x1027, **ATTR_NETWORK_KEY_INDEX** = 0x1028, **ATTR_NEW_DEVICE_NAME** = 0x1029,

  ATTR_NEW_PASSWORD = 0x102a, **ATTR_OOB_DEVICE_PASSWORD** = 0x102c, **ATTR_OS_VERSION** = 0x102d, **ATTR_POWER_LEVEL** = 0x102f,

  ATTR_PSK_CURRENT = 0x1030, **ATTR_PSK_MAX** = 0x1031, **ATTR_PUBLIC_KEY** = 0x1032, **ATTR_RADIO_ENABLE** = 0x1033,

  ATTR_REBOOT = 0x1034, **ATTR_REGISTRAR_CURRENT** = 0x1035, **ATTR_REGISTRAR_ESTABLISHED** = 0x1036, **ATTR_REGISTRAR_LIST** = 0x1037,

  ATTR_REGISTRAR_MAX = 0x1038, **ATTR_REGISTRAR_NONCE** = 0x1039, **ATTR_REQUEST_TYPE** = 0x103a, **ATTR_RESPONSE_TYPE** = 0x103b,

  ATTR_RF_BANDS = 0x103c, **ATTR_R_HASH1** = 0x103d, **ATTR_R_HASH2** = 0x103e, **ATTR_R_SNONCE1** = 0x103f,

  ATTR_R_SNONCE2 = 0x1040, **ATTR_SELECTED_REGISTRAR** = 0x1041, **ATTR_SERIAL_NUMBER** = 0x1042, **ATTR_WPS_STATE** = 0x1044,

  ATTR_SSID = 0x1045, **ATTR_TOTAL_NETWORKS** = 0x1046, **ATTR_UUID_E** = 0x1047, **ATTR_UUID_R** = 0x1048,

  ATTR_VENDOR_EXT = 0x1049, **ATTR_VERSION** = 0x104a, **ATTR_X509_CERT_REQ** = 0x104b, **ATTR_X509_CERT** = 0x104c,

  ATTR_EAP_IDENTITY = 0x104d, **ATTR_MSG_COUNTER** = 0x104e, **ATTR_PUBKEY_HASH** = 0x104f, **ATTR_REKEY_KEY** = 0x1050,

  ATTR_KEY_LIFETIME = 0x1051, **ATTR_PERMITTED_CFG_METHODS** = 0x1052, **ATTR_SELECTED_REGISTRAR_CONFIG_METHODS** = 0x1053, **ATTR_PRIMARY_DEV_TYPE** = 0x1054,

  ATTR_SECONDARY_DEV_TYP_ELIST = 0x1055, **ATTR_PORTABLE_DEV** = 0x1056, **ATTR_AP_SETUP_LOCKED** = 0x1057, **ATTR_APPLICATION_EXT** = 0x1058,

  ATTR_EAP_TYPE = 0x1059, **ATTR_IV** = 0x1060, **ATTR_KEY_PROVIDED_AUTO** = 0x1061, **ATTR_802_1X_ENABLED** = 0x1062,

  ATTR_APPSESSIONKEY = 0x1063, **ATTR_WEPTRANSMITKEY** = 0x1064 }

- enum **wps_dev_password_id** {

  DEV_PW_DEFAULT = 0x0000, **DEV_PW_USER_SPECIFIED** = 0x0001, **DEV_PW_MACHINE_SPECIFIED** = 0x0002, **DEV_PW_REKEY** = 0x0003,

  DEV_PW_PUSHBUTTON = 0x0004, **DEV_PW_REGISTRAR_SPECIFIED** = 0x0005 }
- enum **wps_msg_type** {

  WPS_Beacon = 0x01, **WPS_ProbeRequest** = 0x02, **WPS_ProbeResponse** = 0x03, **WPS_M1** = 0x04,

  WPS_M2 = 0x05, **WPS_M2D** = 0x06, **WPS_M3** = 0x07, **WPS_M4** = 0x08,

  WPS_M5 = 0x09, **WPS_M6** = 0x0a, **WPS_M7** = 0x0b, **WPS_M8** = 0x0c,

  WPS_WSC_ACK = 0x0d, **WPS_WSC_NACK** = 0x0e, **WPS_WSC_DONE** = 0x0f }
- enum **wps_config_error** {

  WPS_CFG_NO_ERROR = 0, **WPS_CFG_OOB_IFACE_READ_ERROR** = 1, **WPS_CFG_DECRYPTION_CRC_FAILURE** = 2, **WPS_CFG_24_CHAN_NOT_SUPPORTED** = 3,

  WPS_CFG_50_CHAN_NOT_SUPPORTED = 4, **WPS_CFG_SIGNAL_TOO_WEAK** = 5, **WPS_CFG_NETWORK_AUTH_FAILURE** = 6, **WPS_CFG_NETWORK_ASSOC_FAILURE** = 7,

  WPS_CFG_NO_DHCP_RESPONSE = 8, **WPS_CFG_FAILED_DHCP_CONFIG** = 9, **WPS_CFG_IP_ADDR_CONFLICT** = 10, **WPS_CFG_NO_CONN_TO_REGISTRAR** = 11,

  WPS_CFG_MULTIPLE_PBC_DETECTED = 12, **WPS_CFG_ROGUE_SUSPECTED** = 13, **WPS_CFG_DEVICE_BUSY** = 14, **WPS_CFG_SETUP_LOCKED** = 15,

  WPS_CFG_MSG_TIMEOUT = 16, **WPS_CFG_REG_SESS_TIMEOUT** = 17, **WPS_CFG_DEV_PASSWORD_AUTH_FAILURE** = 18 }
- enum **wps_state** { **WPS_STATE_NOT_CONFIGURED** = 1, **WPS_STATE_CONFIGURED** = 2 }
- enum **wps_assoc_state** {

  WPS_ASSOC_NOT_ASSOC = 0, **WPS_ASSOC_CONN_SUCCESS** = 1, **WPS_ASSOC_CFG_FAILURE** = 2, **WPS_ASSOC_FAILURE** = 3,

  WPS_ASSOC_IP_FAILURE = 4 }
- enum **wps_dev_categ** {

  WPS_DEV_COMPUTER = 1, **WPS_DEV_INPUT** = 2, **WPS_DEV_PRINTER** = 3, **WPS_DEV_CAMERA** = 4,

  WPS_DEV_STORAGE = 5, **WPS_DEV_NETWORK_INFRA** = 6, **WPS_DEV_DISPLAY** = 7, **WPS_DEV_MULTIMEDIA** = 8,

  WPS_DEV_GAMING = 9, **WPS_DEV_PHONE** = 10 }
- enum **wps_dev_subcateg** {

  WPS_DEV_COMPUTER_PC = 1, **WPS_DEV_COMPUTER_SERVER** = 2, **WPS_DEV_COMPUTER_MEDIA_CENTER** = 3, **WPS_DEV_PRINTER_PRINTER** = 1,

  WPS_DEV_PRINTER_SCANNER = 2, **WPS_DEV_CAMERA_DIGITAL_STILL_CAMERA** = 1, **WPS_DEV_STORAGE_NAS** = 1, **WPS_DEV_NETWORK_INFRA_AP** = 1,

  WPS_DEV_NETWORK_INFRA_ROUTER = 2, **WPS_DEV_NETWORK_INFRA_SWITCH** = 3, **WPS_DEV_DISPLAY_TV** = 1, **WPS_DEV_DISPLAY_PICTURE_FRAME** = 2,

  WPS_DEV_DISPLAY_PROJECTOR = 3, **WPS_DEV_MULTIMEDIA_DAR** = 1, **WPS_DEV_MULTIMEDIA_PVR** = 2, **WPS_DEV_MULTIMEDIA_MCX** = 3,

  WPS_DEV_GAMING_XBOX = 1, **WPS_DEV_GAMING_XBOX360** = 2, **WPS_DEV_GAMING_PLAYSTATION** = 3, **WPS_DEV_PHONE_WINDOWS_MOBILE** = 1 }

- enum **wps_request_type** { **WPS_REQ_ENROLLEE_INFO** = 0, **WPS_REQ_ENROLLEE** = 1, **WPS_REQ_REGISTRAR** = 2, **WPS_REQ_WLAN_MANAGER_REGISTRAR** = 3 }
- enum **wps_response_type** { **WPS_RESP_ENROLLEE_INFO** = 0, **WPS_RESP_ENROLLEE** = 1, **WPS_RESP_REGISTRAR** = 2, **WPS_RESP_AP** = 3 }

15.361.1 Detailed Description

Wi-Fi Protected Setup - message definitions.

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.361.2 Define Documentation

15.361.2.1 #define WPS_AUTH_TYPES

Value:

```
(WPS_AUTH_OPEN | WPS_AUTH_WPAPSK | WPS_AUTH_SHARED | \
                        WPS_AUTH_WPA | WPS_AUTH_WPA2 | WPS_AUTH_WPA2PSK)
```

15.361.2.2 #define WPS_ENCR_TYPES

Value:

```
(WPS_ENCR_NONE | WPS_ENCR_WEP | WPS_ENCR_TKIP | \
                        WPS_ENCR_AES)
```

15.362 src/wps/wps_dev_attr.c File Reference

Wi-Fi Protected Setup - device attributes. `#include "includes.h"`

`#include "common.h"`

`#include "wps_i.h"`

`#include "wps_dev_attr.h"`

Functions

- int **wps_build_primary_dev_type** (struct wps_device_data *dev, struct wpabuf *msg)
- int **wps_build_device_attrs** (struct wps_device_data *dev, struct wpabuf *msg)
- int **wps_build_os_version** (struct wps_device_data *dev, struct wpabuf *msg)
- int **wps_build_rf_bands** (struct wps_device_data *dev, struct wpabuf *msg)
- int **wps_process_device_attrs** (struct wps_device_data *dev, struct wps_parse_attr *attr)
- int **wps_process_os_version** (struct wps_device_data *dev, const u8 *ver)
- int **wps_process_rf_bands** (struct wps_device_data *dev, const u8 *bands)
- void **wps_device_data_dup** (struct wps_device_data *dst, const struct wps_device_data *src)
- void **wps_device_data_free** (struct wps_device_data *dev)

15.362.1 Detailed Description

Wi-Fi Protected Setup - device attributes.

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.363 src/wps/wps_dev_attr.h File Reference

Wi-Fi Protected Setup - device attributes.

Functions

- int **wps_build_device_attrs** (struct wps_device_data *dev, struct wpabuf *msg)
- int **wps_build_os_version** (struct wps_device_data *dev, struct wpabuf *msg)
- int **wps_build_rf_bands** (struct wps_device_data *dev, struct wpabuf *msg)
- int **wps_build_primary_dev_type** (struct wps_device_data *dev, struct wpabuf *msg)
- int **wps_process_device_attrs** (struct wps_device_data *dev, struct wps_parse_attr *attr)
- int **wps_process_os_version** (struct wps_device_data *dev, const u8 *ver)
- int **wps_process_rf_bands** (struct wps_device_data *dev, const u8 *bands)
- void **wps_device_data_dup** (struct wps_device_data *dst, const struct wps_device_data *src)
- void **wps_device_data_free** (struct wps_device_data *dev)

15.363.1 Detailed Description

Wi-Fi Protected Setup - device attributes.

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.364 src/wps/wps_enrollee.c File Reference

Wi-Fi Protected Setup - Enrollee. `#include "includes.h"`

`#include "common.h"`

`#include "sha256.h"`

`#include "wps_i.h"`

`#include "wps_dev_attr.h"`

`#include "crypto.h"`

Functions

- struct wpabuf * **wps_enrollee_get_msg** (struct wps_data *wps, enum wsc_op_code *op_code)
- enum wps_process_res **wps_enrollee_process_msg** (struct wps_data *wps, enum wsc_op_code op_code, const struct wpabuf *msg)

15.364.1 Detailed Description

Wi-Fi Protected Setup - Enrollee.

Copyright

Copyright (c) 2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.365 src/wps/wps_er.c File Reference

Wi-Fi Protected Setup - External Registrar. `#include "includes.h"`

```
#include "common.h"
#include "base64.h"
#include "uuid.h"
#include "eloop.h"
#include "httpread.h"
#include "http_client.h"
#include "http_server.h"
#include "upnp_xml.h"
#include "wps_i.h"
#include "wps_upnp.h"
#include "wps_upnp_i.h"
#include "wps_er.h"
```

Functions

- void **wps_er_ap_add** (struct wps_er *er, const u8 *uuid, struct in_addr *addr, const char *location, int max_age)
- void **wps_er_ap_remove** (struct wps_er *er, struct in_addr *addr)
- struct wps_er * **wps_er_init** (struct wps_context *wps, const char *ifname)
- void **wps_er_refresh** (struct wps_er *er)
- void **wps_er_deinit** (struct wps_er *er)
- void **wps_er_set_sel_reg** (struct wps_er *er, int sel_reg, u16 dev_passwd_id, u16 sel_reg_config_methods)
- int **wps_er_pbc** (struct wps_er *er, const u8 *uuid)
- int **wps_er_learn** (struct wps_er *er, const u8 *uuid, const u8 *pin, size_t pin_len)

15.365.1 Detailed Description

Wi-Fi Protected Setup - External Registrar.

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

15.366 src/wps/wps_er.h File Reference

Wi-Fi Protected Setup - External Registrar.

Data Structures

- struct wps_er_sta
- struct wps_er_ap
- struct wps_er

Functions

- void **wps_er_ap_add** (struct wps_er *er, const u8 *uuid, struct in_addr *addr, const char *location, int max_age)
- void **wps_er_ap_remove** (struct wps_er *er, struct in_addr *addr)
- int **wps_er_ssdp_init** (struct wps_er *er)
- void **wps_er_ssdp_deinit** (struct wps_er *er)
- void **wps_er_send_ssdp_msearch** (struct wps_er *er)

15.366.1 Detailed Description

Wi-Fi Protected Setup - External Registrar.

Copyright

15.367 src/wps/wps_er_ssdp.c File Reference

Wi-Fi Protected Setup - External Registrar (SSDP). `#include "includes.h"`

```
#include "common.h"
#include "uuid.h"
#include "eloop.h"
#include "wps_i.h"
#include "wps_upnp.h"
#include "wps_upnp_i.h"
#include "wps_er.h"
```

Functions

- void **wps_er_send_ssdp_msearch** (struct wps_er *er)
- int **wps_er_ssdp_init** (struct wps_er *er)
- void **wps_er_ssdp_deinit** (struct wps_er *er)

15.367.1 Detailed Description

Wi-Fi Protected Setup - External Registrar (SSDP).

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.368 src/wps/wps_i.h File Reference

Wi-Fi Protected Setup - internal definitions. `#include "wps.h"`

Data Structures

- struct wps_data

  WPS registration protocol data.

- struct wps_parse_attr

Defines

- #define **MAX_CRED_COUNT** 10

Functions

- void **wps_kdf** (const u8 *key, const u8 *label_prefix, size_t label_prefix_len, const char *label, u8 *res, size_t res_len)
- int **wps_derive_keys** (struct wps_data *wps)
- void **wps_derive_psk** (struct wps_data *wps, const u8 *dev_passwd, size_t dev_passwd_len)
- struct wpabuf * **wps_decrypt_encr_settings** (struct wps_data *wps, const u8 *encr, size_t encr_len)
- void **wps_fail_event** (struct wps_context *wps, enum wps_msg_type msg)
- void **wps_success_event** (struct wps_context *wps)
- void **wps_pwd_auth_fail_event** (struct wps_context *wps, int enrollee, int part)
- void **wps_pbc_overlap_event** (struct wps_context *wps)
- void **wps_pbc_timeout_event** (struct wps_context *wps)
- int **wps_parse_msg** (const struct wpabuf *msg, struct wps_parse_attr *attr)
- int **wps_build_public_key** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_req_type** (struct wpabuf *msg, enum wps_request_type type)
- int **wps_build_config_methods** (struct wpabuf *msg, u16 methods)
- int **wps_build_uuid_e** (struct wpabuf *msg, const u8 *uuid)
- int **wps_build_dev_password_id** (struct wpabuf *msg, u16 id)
- int **wps_build_config_error** (struct wpabuf *msg, u16 err)
- int **wps_build_authenticator** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_key_wrap_auth** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_encr_settings** (struct wps_data *wps, struct wpabuf *msg, struct wpabuf *plain)
- int **wps_build_version** (struct wpabuf *msg)
- int **wps_build_msg_type** (struct wpabuf *msg, enum wps_msg_type msg_type)
- int **wps_build_enrollee_nonce** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_registrar_nonce** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_auth_type_flags** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_encr_type_flags** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_conn_type_flags** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_assoc_state** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_build_oob_dev_password** (struct wpabuf *msg, struct wps_context *wps)
- int **wps_process_authenticator** (struct wps_data *wps, const u8 *authenticator, const struct wpabuf *msg)

- int **wps_process_key_wrap_auth** (struct wps_data *wps, struct wpabuf *msg, const u8 *key_wrap_auth)
- int **wps_process_cred** (struct wps_parse_attr *attr, struct wps_credential *cred)
- int **wps_process_ap_settings** (struct wps_parse_attr *attr, struct wps_credential *cred)
- struct wpabuf * **wps_enrollee_get_msg** (struct wps_data *wps, enum wsc_op_code *op_code)
- enum wps_process_res **wps_enrollee_process_msg** (struct wps_data *wps, enum wsc_op_code op_code, const struct wpabuf *msg)
- struct wpabuf * **wps_registrar_get_msg** (struct wps_data *wps, enum wsc_op_code *op_code)
- enum wps_process_res **wps_registrar_process_msg** (struct wps_data *wps, enum wsc_op_code op_code, const struct wpabuf *msg)
- int **wps_build_cred** (struct wps_data *wps, struct wpabuf *msg)
- int **wps_device_store** (struct wps_registrar *reg, struct wps_device_data *dev, const u8 *uuid)
- struct wpabuf * **ndef_parse_wifi** (struct wpabuf *buf)
- struct wpabuf * **ndef_build_wifi** (struct wpabuf *buf)

Variables

- struct oob_device_data **oob_ufd_device_data**
- struct oob_device_data **oob_nfc_device_data**
- struct oob_nfc_device_data **oob_nfc_pn531_device_data**

15.368.1 Detailed Description

Wi-Fi Protected Setup - internal definitions.

Copyright

15.369 src/wps/wps_nfc.c File Reference

NFC routines for Wi-Fi Protected Setup. `#include "includes.h"`

`#include "common.h"`

`#include "wps/wps.h"`

`#include "wps_i.h"`

Data Structures

- struct wps_nfc_data

Variables

- struct oob_device_data **oob_nfc_device_data**

15.369.1 Detailed Description

NFC routines for Wi-Fi Protected Setup.

Copyright

Copyright (c) 2009, Masashi Honma <honma@ictec.co.jp>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.369.2 Variable Documentation

15.369.2.1 struct oob_device_data oob_nfc_device_data

Initial value:

```
{
        .device_name    = NULL,
        .device_path    = NULL,
        .init_func      = init_nfc,
        .read_func      = read_nfc,
        .write_func     = write_nfc,
        .deinit_func    = deinit_nfc,
}
```

15.370 src/wps/wps_nfc_pn531.c File Reference

NFC PN531 routines for Wi-Fi Protected Setup. `#include "includes.h"`

```
#include "common.h"
#include "wps/wps.h"
#include "wps_i.h"
#include "WpsNfcType.h"
#include "WpsNfc.h"
```

Variables

- struct oob_nfc_device_data **oob_nfc_pn531_device_data**

15.370.1 Detailed Description

NFC PN531 routines for Wi-Fi Protected Setup.

Copyright

Copyright (c) 2009, Masashi Honma <honma@ictec.co.jp>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.370.2 Variable Documentation

15.370.2.1 struct oob_nfc_device_data oob_nfc_pn531_device_data

Initial value:

```
 {
        .init_func      = init_nfc_pn531,
        .read_func      = read_nfc_pn531,
        .write_func     = write_nfc_pn531,
        .deinit_func    = deinit_nfc_pn531,
}
```

15.371 src/wps/wps_registrar.c File Reference

Wi-Fi Protected Setup - Registrar. `#include "includes.h"`

```
#include "common.h"
#include "sha256.h"
#include "base64.h"
#include "ieee802_11_defs.h"
#include "eloop.h"
#include "wps_i.h"
#include "wps_dev_attr.h"
#include "wps_upnp.h"
#include "crypto.h"
#include "uuid.h"
```

Data Structures

- struct wps_uuid_pin
- struct wps_pbc_session
- struct wps_registrar_device
- struct wps_registrar

Defines

- #define **WPS_WORKAROUNDS**
- #define **PIN_LOCKED** BIT(0)
- #define **PIN_EXPIRES** BIT(1)
- #define **WPS_STRDUP**(n)

Functions

- int **wps_device_store** (struct wps_registrar *reg, struct wps_device_data *dev, const u8 *uuid)
- struct wps_registrar * wps_registrar_init (struct wps_context *wps, const struct wps_registrar_config *cfg)

  Initialize WPS Registrar data.

- void wps_registrar_deinit (struct wps_registrar *reg)

  Deinitialize WPS Registrar data.

- int wps_registrar_add_pin (struct wps_registrar *reg, const u8 *uuid, const u8 *pin, size_t pin_len, int timeout)

  Configure a new PIN for Registrar.

- int wps_registrar_invalidate_pin (struct wps_registrar *reg, const u8 *uuid)

  Invalidate a PIN for a specific UUID-E.

- int wps_registrar_unlock_pin (struct wps_registrar *reg, const u8 *uuid)

  Unlock a PIN for a specific UUID-E.

- int wps_registrar_button_pushed (struct wps_registrar *reg)

  Notify Registrar that AP button was pushed.

- void wps_registrar_probe_req_rx (struct wps_registrar *reg, const u8 *addr, const struct wpabuf *wps_data)

  Notify Registrar of Probe Request.

- int **wps_build_cred** (struct wps_data *wps, struct wpabuf *msg)
- struct wpabuf * **wps_registrar_get_msg** (struct wps_data *wps, enum wsc_op_code *op_code)
- enum wps_process_res **wps_registrar_process_msg** (struct wps_data *wps, enum wsc_op_code op_code, const struct wpabuf *msg)
- int **wps_registrar_update_ie** (struct wps_registrar *reg)
- int wps_registrar_set_selected_registrar (struct wps_registrar *reg, const struct wpabuf *msg)

  Notification of SetSelectedRegistrar.

- int **wps_registrar_get_info** (struct wps_registrar *reg, const u8 *addr, char *buf, size_t buflen)

15.371.1 Detailed Description

Wi-Fi Protected Setup - Registrar.

Copyright

Copyright (c) 2008-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.371.2 Define Documentation

15.371.2.1 #define WPS_STRDUP(n)

Value:

```
os_free(dst->n); \
        dst->n = src->n ? os_strdup(src->n) : NULL
```

15.371.3 Function Documentation

15.371.3.1 int wps_registrar_add_pin (struct wps_registrar * *reg*, const u8 * *uuid*, const u8 * *pin*, size_t *pin_len*, int *timeout*)

Configure a new PIN for Registrar.

Parameters:

reg Registrar data from wps_registrar_init()

uuid UUID-E or NULL for wildcard (any UUID)

pin PIN (Device Password)

pin_len Length of pin in octets

timeout Time (in seconds) when the PIN will be invalidated; 0 = no timeout

Returns:

0 on success, -1 on failure

15.371.3.2 int wps_registrar_button_pushed (struct wps_registrar * *reg*)

Notify Registrar that AP button was pushed.

Parameters:

reg Registrar data from wps_registrar_init()

Returns:

0 on success, -1 on failure

This function is called on an AP when a push button is pushed to activate PBC mode. The PBC mode will be stopped after walk time (2 minutes) timeout or when a PBC registration is completed.

15.371.3.3 void wps_registrar_deinit (struct wps_registrar * *reg*)

Deinitialize WPS Registrar data.

Parameters:

reg Registrar data from wps_registrar_init()

15.371.3.4 struct wps_registrar* wps_registrar_init (struct wps_context * *wps*, const struct wps_registrar_config * *cfg*) [read]

Initialize WPS Registrar data.

Parameters:

wps Pointer to longterm WPS context

cfg Registrar configuration

Returns:

Pointer to allocated Registrar data or NULL on failure

This function is used to initialize WPS Registrar functionality. It can be used for a single Registrar run (e.g., when run in a supplicant) or multiple runs (e.g., when run as an internal Registrar in an AP). Caller is responsible for freeing the returned data with wps_registrar_deinit() when Registrar functionality is not needed anymore.

15.371.3.5 int wps_registrar_invalidate_pin (struct wps_registrar * *reg*, const u8 * *uuid*)

Invalidate a PIN for a specific UUID-E.

Parameters:

reg Registrar data from wps_registrar_init()

uuid UUID-E

Returns:

0 on success, -1 on failure (e.g., PIN not found)

15.371.3.6 void wps_registrar_probe_req_rx (struct wps_registrar * *reg*, const u8 * *addr*, const struct wpabuf * *wps_data*)

Notify Registrar of Probe Request.

Parameters:

reg Registrar data from wps_registrar_init()

addr MAC address of the Probe Request sender

wps_data WPS IE contents

This function is called on an AP when a Probe Request with WPS IE is received. This is used to track PBC mode use and to detect possible overlap situation with other WPS APs.

15.371.3.7 int wps_registrar_set_selected_registrar (struct wps_registrar * *reg*, const struct wpabuf * *msg*)

Notification of SetSelectedRegistrar.

Parameters:

reg Registrar data from wps_registrar_init()

msg Received message from SetSelectedRegistrar

Returns:

0 on success, -1 on failure

This function is called when an AP receives a SetSelectedRegistrar UPnP message.

15.371.3.8 int wps_registrar_unlock_pin (struct wps_registrar * *reg*, const u8 * *uuid*)

Unlock a PIN for a specific UUID-E.

Parameters:

reg Registrar data from wps_registrar_init()

uuid UUID-E

Returns:

0 on success, -1 on failure

PINs are locked to enforce only one concurrent use. This function unlocks a PIN to allow it to be used again. If the specified PIN was configured using a wildcard UUID, it will be removed instead of allowing multiple uses.

15.372 src/wps/wps_ufd.c File Reference

UFD routines for Wi-Fi Protected Setup. `#include "includes.h"`

```
#include "common.h"
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <dirent.h>
#include "wps/wps.h"
#include "wps/wps_i.h"
```

Data Structures

- struct wps_ufd_data

Defines

- #define **UFD_DIR1** "%s/SMRTNTKY"
- #define **UFD_DIR2** UFD_DIR1 "/WFAWSC"
- #define **UFD_FILE** UFD_DIR2 "/%s"

Variables

- struct oob_device_data **oob_ufd_device_data**

15.372.1 Detailed Description

UFD routines for Wi-Fi Protected Setup.

Copyright

Copyright (c) 2009, Masashi Honma <honma@ictec.co.jp>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.372.2 Variable Documentation

15.372.2.1 struct oob_device_data oob_ufd_device_data

Initial value:

```
 {
        .device_name    = NULL,
        .device_path    = NULL,
        .init_func      = init_ufd,
        .read_func      = read_ufd,
        .write_func     = write_ufd,
        .deinit_func    = deinit_ufd,
}
```

15.373 src/wps/wps_upnp.c File Reference

UPnP WPS Device. `#include "includes.h"`

```
#include <assert.h>
#include <net/if.h>
#include <netdb.h>
#include <sys/ioctl.h>
#include "common.h"
#include "uuid.h"
#include "base64.h"
#include "wps.h"
#include "wps_i.h"
#include "wps_upnp.h"
#include "wps_upnp_i.h"
```

Defines

- #define **NO_DOMAIN_NAME_RESOLUTION** 1
- #define **MAX_SUBSCRIPTIONS** 4
- #define **MAX_ADDR_PER_SUBSCRIPTION** 8

Functions

- void **format_date** (struct wpabuf *buf)
- int **send_wpabuf** (int fd, struct wpabuf *buf)
- void **subscription_unlink** (struct subscription *s)
- void **subscription_destroy** (struct subscription *s)
- struct subscription * **subscription_find** (struct upnp_wps_device_sm *sm, const u8 uuid[UUID_-LEN])
- struct subscription * subscription_start (struct upnp_wps_device_sm *sm, const char *callback_-urls)

  Remember a UPnP control point to send events to.

- struct subscription * **subscription_renew** (struct upnp_wps_device_sm *sm, const u8 uuid[UUID_-LEN])
- int upnp_wps_device_send_wlan_event (struct upnp_wps_device_sm *sm, const u8 from_mac_-addr[ETH_ALEN], enum upnp_wps_wlanevent_type ev_type, const struct wpabuf *msg)

  Event notification.

- int get_netif_info (const char *net_if, unsigned *ip_addr, char **ip_addr_text, u8 mac[ETH_-ALEN], char **mac_addr_text)

  Get hw and IP addresses for network device.

- void upnp_wps_device_stop (struct upnp_wps_device_sm *sm)

  Stop WPS UPnP operations on an interface.

- int upnp_wps_device_start (struct upnp_wps_device_sm *sm, char *net_if)

  Start WPS UPnP operations on an interface.

- void upnp_wps_device_deinit (struct upnp_wps_device_sm *sm)

  Deinitialize WPS UPnP.

- struct upnp_wps_device_sm * upnp_wps_device_init (struct upnp_wps_device_ctx *ctx, struct wps_context *wps, void *priv)

  Initialize WPS UPnP.

- int upnp_wps_subscribers (struct upnp_wps_device_sm *sm)

  Check whether there are any event subscribers.

15.373.1 Detailed Description

UPnP WPS Device.

Copyright

Copyright (c) 2000-2003 Intel Corporation Copyright (c) 2006-2007 Sony Corporation Copyright (c) 2008-2009 Atheros Communications Copyright (c) 2009, Jouni Malinen <j@w1.fi>

See below for more details on licensing and code history.

15.373.2 Function Documentation

15.373.2.1 int get_netif_info (const char * *net_if*, unsigned * *ip_addr*, char ** *ip_addr_text*, u8 *mac*[ETH_ALEN], char ** *mac_addr_text*)

Get hw and IP addresses for network device.

Parameters:

net_if Selected network interface name

ip_addr Buffer for returning IP address in network byte order

ip_addr_text Buffer for returning a pointer to allocated IP address text

mac Buffer for returning MAC address

mac_addr_text Buffer for returning allocated MAC address text

Returns:

0 on success, -1 on failure

15.373.2.2 struct subscription* subscription_start (struct upnp_wps_device_sm * *sm*, const char * *callback_urls*) [read]

Remember a UPnP control point to send events to.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

callback_urls Callback URLs

Returns:

NULL on error, or pointer to new subscription structure.

15.373.2.3 void upnp_wps_device_deinit (struct upnp_wps_device_sm * *sm*)

Deinitialize WPS UPnP.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

15.373.2.4 struct upnp_wps_device_sm* upnp_wps_device_init (struct upnp_wps_device_ctx * *ctx*, struct wps_context * *wps*, void * *priv*) [read]

Initialize WPS UPnP.

Parameters:

ctx callback table; we must eventually free it

wps Pointer to longterm WPS context

priv External context data that will be used in callbacks

Returns:

WPS UPnP state or NULL on failure

15.373.2.5 int upnp_wps_device_send_wlan_event (struct upnp_wps_device_sm * *sm*, const u8 *from_mac_addr*[ETH_ALEN], enum upnp_wps_wlanevent_type *ev_type*, const struct wpabuf * *msg*)

Event notification.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

from_mac_addr Source (Enrollee) MAC address for the event

ev_type Event type

msg Event data

Returns:

0 on success, -1 on failure

Tell external Registrars (UPnP control points) that something happened. In particular, events include WPS messages from clients that are proxied to external Registrars.

15.373.2.6 int upnp_wps_device_start (struct upnp_wps_device_sm * *sm*, char * *net_if*)

Start WPS UPnP operations on an interface.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

net_if Selected network interface name

Returns:

0 on success, -1 on failure

15.373.2.7 void upnp_wps_device_stop (struct upnp_wps_device_sm * *sm*)

Stop WPS UPnP operations on an interface.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

15.373.2.8 int upnp_wps_subscribers (struct upnp_wps_device_sm * *sm*)

Check whether there are any event subscribers.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

Returns:

0 if no subscribers, 1 if subscribers

15.374 src/wps/wps_upnp.h File Reference

UPnP WPS Device.

Data Structures

- struct upnp_wps_peer
- struct upnp_wps_device_ctx

Enumerations

- enum **upnp_wps_wlanevent_type** { **UPNP_WPS_WLANEVENT_TYPE_PROBE** = 1, **UPNP_WPS_WLANEVENT_TYPE_EAP** = 2 }

Functions

- struct upnp_wps_device_sm * upnp_wps_device_init (struct upnp_wps_device_ctx *ctx, struct wps_context *wps, void *priv)

  Initialize WPS UPnP.

- void upnp_wps_device_deinit (struct upnp_wps_device_sm *sm)

  Deinitialize WPS UPnP.

- int upnp_wps_device_start (struct upnp_wps_device_sm *sm, char *net_if)

  Start WPS UPnP operations on an interface.

- void upnp_wps_device_stop (struct upnp_wps_device_sm *sm)

  Stop WPS UPnP operations on an interface.

- int upnp_wps_device_send_wlan_event (struct upnp_wps_device_sm *sm, const u8 from_mac_addr[ETH_ALEN], enum upnp_wps_wlanevent_type ev_type, const struct wpabuf *msg)

  Event notification.

- int upnp_wps_subscribers (struct upnp_wps_device_sm *sm)

  Check whether there are any event subscribers.

15.374.1 Detailed Description

UPnP WPS Device.

Copyright

Copyright (c) 2000-2003 Intel Corporation Copyright (c) 2006-2007 Sony Corporation Copyright (c) 2008-2009 Atheros Communications Copyright (c) 2009, Jouni Malinen <j@w1.fi>

See wps_upnp.c for more details on licensing and code history.

15.374.2 Function Documentation

15.374.2.1 void upnp_wps_device_deinit (struct upnp_wps_device_sm * *sm*)

Deinitialize WPS UPnP.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

15.374.2.2 struct upnp_wps_device_sm* upnp_wps_device_init (struct upnp_wps_device_ctx * *ctx*, struct wps_context * *wps*, void * *priv*) [read]

Initialize WPS UPnP.

Parameters:

ctx callback table; we must eventually free it

wps Pointer to longterm WPS context

priv External context data that will be used in callbacks

Returns:

WPS UPnP state or NULL on failure

15.374.2.3 int upnp_wps_device_send_wlan_event (struct upnp_wps_device_sm * *sm*, const u8 *from_mac_addr*[ETH_ALEN], enum upnp_wps_wlanevent_type *ev_type*, const struct wpabuf * *msg*)

Event notification.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

from_mac_addr Source (Enrollee) MAC address for the event

ev_type Event type

msg Event data

Returns:

0 on success, -1 on failure

Tell external Registrars (UPnP control points) that something happened. In particular, events include WPS messages from clients that are proxied to external Registrars.

15.374.2.4 int upnp_wps_device_start (struct upnp_wps_device_sm * *sm*, char * *net_if*)

Start WPS UPnP operations on an interface.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

net_if Selected network interface name

Returns:

0 on success, -1 on failure

15.374.2.5 void upnp_wps_device_stop (struct upnp_wps_device_sm * *sm*)

Stop WPS UPnP operations on an interface.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

15.374.2.6 int upnp_wps_subscribers (struct upnp_wps_device_sm * *sm*)

Check whether there are any event subscribers.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

Returns:

0 if no subscribers, 1 if subscribers

15.375 src/wps/wps_upnp_event.c File Reference

UPnP WPS Device - Event processing. `#include "includes.h"`

```
#include <assert.h>
#include "common.h"
#include "eloop.h"
#include "uuid.h"
#include "http_client.h"
#include "wps_defs.h"
#include "wps_upnp.h"
#include "wps_upnp_i.h"
```

Data Structures

- struct wps_event_

Defines

- #define **MAX_EVENTS_QUEUED** 20
- #define **EVENT_TIMEOUT_SEC** 30
- #define **EVENT_DELAY_SECONDS** 0
- #define **EVENT_DELAY_MSEC** 0

Functions

- void **event_delete_all** (struct subscription *s)
- void **event_send_all_later** (struct upnp_wps_device_sm *sm)
- void **event_send_stop_all** (struct upnp_wps_device_sm *sm)
- int event_add (struct subscription *s, const struct wpabuf *data)

  Add a new event to a queue.

15.375.1 Detailed Description

UPnP WPS Device - Event processing.

Copyright

Copyright (c) 2000-2003 Intel Corporation Copyright (c) 2006-2007 Sony Corporation Copyright (c) 2008-2009 Atheros Communications Copyright (c) 2009, Jouni Malinen <j@w1.fi>

See wps_upnp.c for more details on licensing and code history.

15.375.2 Function Documentation

15.375.2.1 int event_add (struct subscription * *s*, const struct wpabuf * *data*)

Add a new event to a queue.

Parameters:

s Subscription

data Event data (is copied; caller retains ownership)

Returns:

0 on success, 1 on error

15.376 src/wps/wps_upnp_i.h File Reference

UPnP for WPS / internal definitions. `#include "http.h"`

Data Structures

- struct advertisement_state_machine
- struct subscr_addr
- struct subscription
- struct upnp_wps_device_sm

Defines

- #define **UPNP_MULTICAST_ADDRESS** "239.255.255.250"
- #define **UPNP_MULTICAST_PORT** 1900
- #define **UPNP_SUBSCRIBE_SEC_MIN** 1800
- #define **UPNP_SUBSCRIBE_SEC** (UPNP_SUBSCRIBE_SEC_MIN + 1)
- #define **UPNP_WPS_DEVICE_XML_FILE** "wps_device.xml"
- #define **UPNP_WPS_SCPD_XML_FILE** "wps_scpd.xml"
- #define **UPNP_WPS_DEVICE_CONTROL_FILE** "wps_control"
- #define **UPNP_WPS_DEVICE_EVENT_FILE** "wps_event"
- #define **MULTICAST_MAX_READ** 1600

Enumerations

- enum **advertisement_type_enum** { **ADVERTISE_UP** = 0, **ADVERTISE_DOWN** = 1, **MSEARCH_REPLY** = 2 }

Functions

- void **format_date** (struct wpabuf *buf)
- struct subscription * subscription_start (struct upnp_wps_device_sm *sm, const char *callback_-urls)

  Remember a UPnP control point to send events to.

- struct subscription * **subscription_renew** (struct upnp_wps_device_sm *sm, const u8 uuid[UUID_-LEN])
- void **subscription_unlink** (struct subscription *s)
- void **subscription_destroy** (struct subscription *s)
- struct subscription * **subscription_find** (struct upnp_wps_device_sm *sm, const u8 uuid[UUID_-LEN])
- int **send_wpabuf** (int fd, struct wpabuf *buf)
- int get_netif_info (const char *net_if, unsigned *ip_addr, char **ip_addr_text, u8 mac[ETH_-ALEN], char **mac_addr_text)

  Get hw and IP addresses for network device.

- void msearchreply_state_machine_stop (struct advertisement_state_machine *a)

  Stop M-SEARCH reply state machine.

- int advertisement_state_machine_start (struct upnp_wps_device_sm *sm)

  Start SSDP advertisements.

- void advertisement_state_machine_stop (struct upnp_wps_device_sm *sm, int send_byebye)

  Stop SSDP advertisements.

- void ssdp_listener_stop (struct upnp_wps_device_sm *sm)

  Stop SSDP listered.

- int ssdp_listener_start (struct upnp_wps_device_sm *sm)

  Set up for receiving discovery (UDP) packets.

- int **ssdp_listener_open** (void)
- int add_ssdp_network (const char *net_if)

  Add routing entry for SSDP.

- int **ssdp_open_multicast_sock** (u32 ip_addr)
- int ssdp_open_multicast (struct upnp_wps_device_sm *sm)

  Open socket for sending multicast SSDP messages.

- int **web_listener_start** (struct upnp_wps_device_sm *sm)
- void **web_listener_stop** (struct upnp_wps_device_sm *sm)
- int event_add (struct subscription *s, const struct wpabuf *data)

  Add a new event to a queue.

- void **event_delete_all** (struct subscription *s)
- void **event_send_all_later** (struct upnp_wps_device_sm *sm)
- void **event_send_stop_all** (struct upnp_wps_device_sm *sm)

15.376.1 Detailed Description

UPnP for WPS / internal definitions.

Copyright

Copyright (c) 2000-2003 Intel Corporation Copyright (c) 2006-2007 Sony Corporation Copyright (c) 2008-2009 Atheros Communications Copyright (c) 2009, Jouni Malinen <j@w1.fi>

See wps_upnp.c for more details on licensing and code history.

15.376.2 Function Documentation

15.376.2.1 int add_ssdp_network (const char * *net_if*)

Add routing entry for SSDP.

Parameters:

net_if Selected network interface name

Returns:

0 on success, -1 on failure

This function assures that the multicast address will be properly handled by Linux networking code (by a modification to routing tables). This must be done per network interface. It really only needs to be done once after booting up, but it does not hurt to call this more frequently "to be safe".

15.376.2.2 int advertisement_state_machine_start (struct upnp_wps_device_sm * *sm*)

Start SSDP advertisements.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

Returns:

0 on success, -1 on failure

15.376.2.3 void advertisement_state_machine_stop (struct upnp_wps_device_sm * *sm*, int *send_byebye*)

Stop SSDP advertisements.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

send_byebye Send byebye advertisement messages immediately

15.376.2.4 int event_add (struct subscription * *s*, const struct wpabuf * *data*)

Add a new event to a queue.

Parameters:

s Subscription

data Event data (is copied; caller retains ownership)

Returns:

0 on success, 1 on error

15.376.2.5 int get_netif_info (const char * *net_if*, unsigned * *ip_addr*, char ** *ip_addr_text*, u8 *mac*[ETH_ALEN], char ** *mac_addr_text*)

Get hw and IP addresses for network device.

Parameters:

net_if Selected network interface name

ip_addr Buffer for returning IP address in network byte order

ip_addr_text Buffer for returning a pointer to allocated IP address text

mac Buffer for returning MAC address

mac_addr_text Buffer for returning allocated MAC address text

Returns:

0 on success, -1 on failure

15.376.2.6 void msearchreply_state_machine_stop (struct advertisement_state_machine * *a*)

Stop M-SEARCH reply state machine.

Parameters:

a Selected advertisement/reply state

15.376.2.7 int ssdp_listener_start (struct upnp_wps_device_sm * *sm*)

Set up for receiving discovery (UDP) packets.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

Returns:

0 on success, -1 on failure

The SSDP listener is stopped by calling ssdp_listener_stop().

15.376.2.8 void ssdp_listener_stop (struct upnp_wps_device_sm * *sm*)

Stop SSDP listered.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

This function stops the SSDP listener that was started by calling ssdp_listener_start().

15.376.2.9 int ssdp_open_multicast (struct upnp_wps_device_sm * *sm*)

Open socket for sending multicast SSDP messages.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

Returns:

0 on success, -1 on failure

15.376.2.10 struct subscription* subscription_start (struct upnp_wps_device_sm * *sm*, const char * *callback_urls*) [read]

Remember a UPnP control point to send events to.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

callback_urls Callback URLs

Returns:

NULL on error, or pointer to new subscription structure.

15.377 src/wps/wps_upnp_ssdp.c File Reference

UPnP SSDP for WPS. `#include "includes.h"`

```
#include <fcntl.h>
#include <sys/ioctl.h>
#include <net/route.h>
#include "common.h"
#include "uuid.h"
#include "eloop.h"
#include "wps.h"
#include "wps_upnp.h"
#include "wps_upnp_i.h"
```

Defines

- #define **UPNP_CACHE_SEC** (UPNP_CACHE_SEC_MIN + 1)
- #define **UPNP_CACHE_SEC_MIN** 1800
- #define **UPNP_ADVERTISE_REPEAT** 2
- #define **MAX_MSEARCH** 20
- #define **SSDP_TARGET** "239.0.0.0"
- #define **SSDP_NETMASK** "255.0.0.0"

Functions

- void advertisement_state_machine_stop (struct upnp_wps_device_sm *sm, int send_byebye)
  Stop SSDP advertisements.
- int advertisement_state_machine_start (struct upnp_wps_device_sm *sm)
  Start SSDP advertisements.
- void msearchreply_state_machine_stop (struct advertisement_state_machine *a)
  Stop M-SEARCH reply state machine.
- void ssdp_listener_stop (struct upnp_wps_device_sm *sm)
  Stop SSDP listered.
- int **ssdp_listener_open** (void)
- int ssdp_listener_start (struct upnp_wps_device_sm *sm)
  Set up for receiving discovery (UDP) packets.
- int add_ssdp_network (const char *net_if)
  Add routing entry for SSDP.
- int **ssdp_open_multicast_sock** (u32 ip_addr)
- int ssdp_open_multicast (struct upnp_wps_device_sm *sm)
  Open socket for sending multicast SSDP messages.

15.377.1 Detailed Description

UPnP SSDP for WPS.

Copyright

Copyright (c) 2000-2003 Intel Corporation Copyright (c) 2006-2007 Sony Corporation Copyright (c) 2008-2009 Atheros Communications Copyright (c) 2009, Jouni Malinen <j@w1.fi>

See wps_upnp.c for more details on licensing and code history.

15.377.2 Function Documentation

15.377.2.1 int add_ssdp_network (const char * *net_if*)

Add routing entry for SSDP.

Parameters:

net_if Selected network interface name

Returns:

0 on success, -1 on failure

This function assures that the multicast address will be properly handled by Linux networking code (by a modification to routing tables). This must be done per network interface. It really only needs to be done once after booting up, but it does not hurt to call this more frequently "to be safe".

15.377.2.2 int advertisement_state_machine_start (struct upnp_wps_device_sm * *sm*)

Start SSDP advertisements.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

Returns:

0 on success, -1 on failure

15.377.2.3 void advertisement_state_machine_stop (struct upnp_wps_device_sm * *sm*, int *send_byebye*)

Stop SSDP advertisements.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

send_byebye Send byebye advertisement messages immediately

15.377.2.4 void msearchreply_state_machine_stop (struct advertisement_state_machine * *a*)

Stop M-SEARCH reply state machine.

Parameters:

a Selected advertisement/reply state

15.377.2.5 int ssdp_listener_start (struct upnp_wps_device_sm * *sm*)

Set up for receiving discovery (UDP) packets.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

Returns:

0 on success, -1 on failure

The SSDP listener is stopped by calling ssdp_listener_stop().

15.377.2.6 void ssdp_listener_stop (struct upnp_wps_device_sm * *sm*)

Stop SSDP listered.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

This function stops the SSDP listener that was started by calling ssdp_listener_start().

15.377.2.7 int ssdp_open_multicast (struct upnp_wps_device_sm * *sm*)

Open socket for sending multicast SSDP messages.

Parameters:

sm WPS UPnP state machine from upnp_wps_device_init()

Returns:

0 on success, -1 on failure

15.378 src/wps/wps_upnp_web.c File Reference

UPnP WPS Device - Web connections. `#include "includes.h"`

```
#include "common.h"
#include "base64.h"
#include "uuid.h"
#include "httpread.h"
#include "http_server.h"
#include "wps_i.h"
#include "wps_upnp.h"
#include "wps_upnp_i.h"
#include "upnp_xml.h"
```

Defines

- #define **WEB_CONNECTION_TIMEOUT_SEC** 30
- #define **WEB_CONNECTION_MAX_READ** 8000
- #define **MAX_WEB_CONNECTIONS** 10

Functions

- void **web_listener_stop** (struct upnp_wps_device_sm *sm)
- int **web_listener_start** (struct upnp_wps_device_sm *sm)

15.378.1 Detailed Description

UPnP WPS Device - Web connections.

Copyright

Copyright (c) 2000-2003 Intel Corporation Copyright (c) 2006-2007 Sony Corporation Copyright (c) 2008-2009 Atheros Communications Copyright (c) 2009, Jouni Malinen <j@w1.fi>

See wps_upnp.c for more details on licensing and code history.

15.379 wpa_supplicant/ap.c File Reference

WPA Supplicant - Basic AP mode support routines. `#include "includes.h"`

```
#include "common.h"
#include "../hostapd/hostapd.h"
#include "../hostapd/config.h"
#include "../hostapd/wps_hostapd.h"
#include "../hostapd/ctrl_iface_ap.h"
#include "eap_common/eap_defs.h"
#include "eap_server/eap_methods.h"
#include "eap_common/eap_wsc_common.h"
#include "wps/wps.h"
#include "config_ssid.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "ap.h"
```

Data Structures

- struct ap_driver_data

Functions

- int **hostapd_for_each_interface** (int(*cb)(struct hostapd_iface *iface, void *ctx), void *ctx)
- int **hostapd_ctrl_iface_init** (struct hostapd_data *hapd)
- void **hostapd_ctrl_iface_deinit** (struct hostapd_data *hapd)
- int **wpa_supplicant_create_ap** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpa_supplicant_ap_deinit** (struct wpa_supplicant *wpa_s)
- void **ap_tx_status** (void *ctx, const u8 *addr, const u8 *buf, size_t len, int ack)
- void **ap_rx_from_unknown_sta** (void *ctx, struct ieee80211_hdr *hdr, size_t len)
- void **wpa_supplicant_ap_rx_eapol** (struct wpa_supplicant *wpa_s, const u8 *src_addr, const u8 *buf, size_t len)
- int **ap_ctrl_iface_sta_first** (struct wpa_supplicant *wpa_s, char *buf, size_t buflen)
- int **ap_ctrl_iface_sta** (struct wpa_supplicant *wpa_s, const char *txtaddr, char *buf, size_t buflen)
- int **ap_ctrl_iface_sta_next** (struct wpa_supplicant *wpa_s, const char *txtaddr, char *buf, size_t buflen)
- int **ap_ctrl_iface_wpa_get_status** (struct wpa_supplicant *wpa_s, char *buf, size_t buflen, int verbose)

Variables

- struct wpa_driver_ops **ap_driver_ops**
- struct wpa_driver_ops * **wpa_drivers** [ ]

15.379.1 Detailed Description

WPA Supplicant - Basic AP mode support routines.

Copyright

Copyright (c) 2003-2009, Jouni Malinen <j@w1.fi> Copyright (c) 2009, Atheros Communications

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.379.2 Variable Documentation

15.379.2.1 struct wpa_driver_ops ap_driver_ops

Initial value:

```
{
        .name = "wpa_supplicant",
        .hapd_init = ap_driver_init,
        .hapd_deinit = ap_driver_deinit,
        .send_ether = ap_driver_send_ether,
        .set_key = ap_driver_set_key,
        .get_seqnum = ap_driver_get_seqnum,
        .flush = ap_driver_flush,
        .read_sta_data = ap_driver_read_sta_data,
        .sta_set_flags = ap_driver_sta_set_flags,
        .sta_deauth = ap_driver_sta_deauth,
        .sta_disassoc = ap_driver_sta_disassoc,
        .sta_remove = ap_driver_sta_remove,
        .send_mlme = ap_driver_send_mlme,
        .sta_add = ap_driver_sta_add,
        .get_inact_sec = ap_driver_get_inact_sec,
        .set_freq = ap_driver_set_freq,
        .set_beacon = ap_driver_set_beacon,
        .set_cts_protect = ap_driver_set_cts_protect,
        .set_preamble = ap_driver_set_preamble,
        .set_short_slot_time = ap_driver_set_short_slot_time,
        .set_tx_queue_params = ap_driver_set_tx_queue_params,
        .get_hw_feature_data = ap_driver_get_hw_feature_data,
        .hapd_send_eapol = ap_driver_hapd_send_eapol,
}
```

15.380 wpa_supplicant/ap.h File Reference

WPA Supplicant - Basic AP mode support routines.

Functions

- int **wpa_supplicant_create_ap** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpa_supplicant_ap_deinit** (struct wpa_supplicant *wpa_s)
- void **wpa_supplicant_ap_rx_eapol** (struct wpa_supplicant *wpa_s, const u8 *src_addr, const u8 *buf, size_t len)
- int **wpa_supplicant_ap_wps_pbc** (struct wpa_supplicant *wpa_s, const u8 *bssid)
- int **wpa_supplicant_ap_wps_pin** (struct wpa_supplicant *wpa_s, const u8 *bssid, const char *pin, char *buf, size_t buflen)
- int **ap_ctrl_iface_sta_first** (struct wpa_supplicant *wpa_s, char *buf, size_t buflen)
- int **ap_ctrl_iface_sta** (struct wpa_supplicant *wpa_s, const char *txtaddr, char *buf, size_t buflen)
- int **ap_ctrl_iface_sta_next** (struct wpa_supplicant *wpa_s, const char *txtaddr, char *buf, size_t buflen)
- int **ap_ctrl_iface_wpa_get_status** (struct wpa_supplicant *wpa_s, char *buf, size_t buflen, int verbose)

15.380.1 Detailed Description

WPA Supplicant - Basic AP mode support routines.

Copyright

Copyright (c) 2003-2009, Jouni Malinen <j@w1.fi> Copyright (c) 2009, Atheros Communications

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.381 wpa_supplicant/bgscan.c File Reference

WPA Supplicant - background scan and roaming interface. `#include "includes.h"`

```
#include "common.h"
#include "wpa_supplicant_i.h"
#include "config_ssid.h"
#include "bgscan.h"
```

Functions

- int **bgscan_init** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **bgscan_deinit** (struct wpa_supplicant *wpa_s)
- int **bgscan_notify_scan** (struct wpa_supplicant *wpa_s)
- void **bgscan_notify_beacon_loss** (struct wpa_supplicant *wpa_s)
- void **bgscan_notify_signal_change** (struct wpa_supplicant *wpa_s)

15.381.1 Detailed Description

WPA Supplicant - background scan and roaming interface.

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.382 wpa_supplicant/bgscan.h File Reference

WPA Supplicant - background scan and roaming interface.

Data Structures

- struct bgscan_ops

15.382.1 Detailed Description

WPA Supplicant - background scan and roaming interface.

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.383 wpa_supplicant/bgscan_simple.c File Reference

WPA Supplicant - background scan and roaming module: simple. `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "drivers/driver.h"
#include "config_ssid.h"
#include "wpa_supplicant_i.h"
#include "bgscan.h"
```

Data Structures

- struct bgscan_simple_data

Variables

- struct bgscan_ops **bgscan_simple_ops**

15.383.1 Detailed Description

WPA Supplicant - background scan and roaming module: simple.

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.383.2 Variable Documentation

15.383.2.1 struct bgscan_ops bgscan_simple_ops

Initial value:

```
{
        .name = "simple",
        .init = bgscan_simple_init,
        .deinit = bgscan_simple_deinit,
        .notify_scan = bgscan_simple_notify_scan,
        .notify_beacon_loss = bgscan_simple_notify_beacon_loss,
        .notify_signal_change = bgscan_simple_notify_signal_change,
}
```

15.384 wpa_supplicant/blacklist.c File Reference

wpa_supplicant - Temporary BSSID blacklist `#include "includes.h"`

`#include "common.h"`

`#include "wpa_supplicant_i.h"`

`#include "blacklist.h"`

Functions

- struct 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.
- int wpa_blacklist_del (struct wpa_supplicant *wpa_s, const u8 *bssid)
  Remove an BSSID from the blacklist.
- void wpa_blacklist_clear (struct wpa_supplicant *wpa_s)
  Clear the blacklist of all entries.

15.384.1 Detailed Description

wpa_supplicant - Temporary BSSID blacklist

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.384.2 Function Documentation

15.384.2.1 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.

15.384.2.2 void wpa_blacklist_clear (struct wpa_supplicant * *wpa_s*)

Clear the blacklist of all entries.

Parameters:

wpa_s Pointer to wpa_supplicant data

15.384.2.3 int wpa_blacklist_del (struct wpa_supplicant * *wpa_s*, const u8 * *bssid*)

Remove an BSSID from the blacklist.

Parameters:

wpa_s Pointer to wpa_supplicant data

bssid BSSID to be removed from the blacklist

Returns:

0 on success, -1 on failure

15.384.2.4 struct wpa_blacklist* wpa_blacklist_get (struct wpa_supplicant * *wpa_s*, const u8 * *bssid*) [read]

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

15.385 wpa_supplicant/blacklist.h File Reference

wpa_supplicant - Temporary BSSID blacklist

Data Structures

- struct wpa_blacklist

Functions

- struct 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.
- int wpa_blacklist_del (struct wpa_supplicant *wpa_s, const u8 *bssid)
  Remove an BSSID from the blacklist.
- void wpa_blacklist_clear (struct wpa_supplicant *wpa_s)
  Clear the blacklist of all entries.

15.385.1 Detailed Description

wpa_supplicant - Temporary BSSID blacklist

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.385.2 Function Documentation

15.385.2.1 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.

15.385.2.2 void wpa_blacklist_clear (struct wpa_supplicant * *wpa_s*)

Clear the blacklist of all entries.

Parameters:

wpa_s Pointer to wpa_supplicant data

15.385.2.3 int wpa_blacklist_del (struct wpa_supplicant * *wpa_s*, const u8 * *bssid*)

Remove an BSSID from the blacklist.

Parameters:

wpa_s Pointer to wpa_supplicant data

bssid BSSID to be removed from the blacklist

Returns:

0 on success, -1 on failure

15.385.2.4 struct wpa_blacklist* wpa_blacklist_get (struct wpa_supplicant * *wpa_s*, const u8 * *bssid*) [read]

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

15.386 wpa_supplicant/config_file.c File Reference

WPA Supplicant / Configuration backend: text file. `#include "includes.h"`

```
#include "common.h"
#include "config.h"
#include "base64.h"
#include "uuid.h"
#include "eap_peer/eap_methods.h"
```

Data Structures

- struct global_parse_data

Defines

- #define **OFFSET**(v) ((void *) &((struct wpa_config *) 0)->v)
- #define **FUNC**(f) #f, wpa_config_process_ ## f, OFFSET(f), NULL, NULL
- #define **FUNC_NO_VAR**(f) #f, wpa_config_process_ ## f, NULL, NULL, NULL
- #define **_INT**(f) #f, wpa_config_parse_int, OFFSET(f)
- #define **INT**(f) _INT(f), NULL, NULL
- #define **INT_RANGE**(f, min, max) _INT(f), (void *) min, (void *) max
- #define **_STR**(f) #f, wpa_config_parse_str, OFFSET(f)
- #define **STR**(f) _STR(f), NULL, NULL
- #define **STR_RANGE**(f, min, max) _STR(f), (void *) min, (void *) max
- #define **NUM_GLOBAL_FIELDS** (sizeof(global_fields) / sizeof(global_fields[0]))
- #define **STR**(t) write_str(f, #t, ssid)
- #define **INT**(t) write_int(f, #t, ssid->t, 0)
- #define **INTe**(t) write_int(f, #t, ssid->eap.t, 0)
- #define **INT_DEF**(t, def) write_int(f, #t, ssid->t, def)
- #define **INT_DEFe**(t, def) write_int(f, #t, ssid->eap.t, def)

Functions

- struct 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.

15.386.1 Detailed Description

WPA Supplicant / Configuration backend: text file.

Copyright

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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.

15.386.2 Function Documentation

15.386.2.1 struct wpa_config* wpa_config_read (const char * *name*) [read]

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.

15.386.2.2 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.

15.387 wpa_supplicant/config_none.c File Reference

WPA Supplicant / Configuration backend: empty starting point. `#include "includes.h"`

`#include "common.h"`

`#include "config.h"`

`#include "base64.h"`

Functions

- struct 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.

15.387.1 Detailed Description

WPA Supplicant / Configuration backend: empty starting point.

Copyright

Copyright (c) 2003-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements dummy example of a configuration backend. None of the functions are actually implemented so this can be used as a simple compilation test or a starting point for a new configuration backend.

15.387.2 Function Documentation

15.387.2.1 struct wpa_config* wpa_config_read (const char * *name*) [read]

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.

15.387.2.2 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.

15.388 wpa_supplicant/config_ssid.h File Reference

WPA Supplicant / Network configuration structures. `#include "defs.h"`

`#include "eap_peer/eap_config.h"`

Data Structures

- struct wpa_ssid

  Network configuration data.

Defines

- #define **MAX_SSID_LEN** 32
- #define **DEFAULT_EAP_WORKAROUND** ((unsigned int) -1)
- #define **DEFAULT_EAPOL_FLAGS**
- #define **DEFAULT_PROTO** (WPA_PROTO_WPA | WPA_PROTO_RSN)
- #define **DEFAULT_KEY_MGMT** (WPA_KEY_MGMT_PSK | WPA_KEY_MGMT_-IEEE8021X)
- #define **DEFAULT_PAIRWISE** (WPA_CIPHER_CCMP | WPA_CIPHER_TKIP)
- #define **DEFAULT_GROUP**
- #define **DEFAULT_FRAGMENT_SIZE** 1398
- #define **EAPOL_FLAG_REQUIRE_KEY_UNICAST** BIT(0)
- #define **EAPOL_FLAG_REQUIRE_KEY_BROADCAST** BIT(1)
- #define **NUM_WEP_KEYS** 4
- #define **MAX_WEP_KEY_LEN** 16

15.388.1 Detailed Description

WPA Supplicant / Network configuration structures.

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.388.2 Define Documentation

15.388.2.1 #define DEFAULT_EAPOL_FLAGS

Value:

```
(EAPOL_FLAG_REQUIRE_KEY_UNICAST | \
                              EAPOL_FLAG_REQUIRE_KEY_BROADCAST)
```

15.388.2.2 #define DEFAULT_GROUP

Value:

```
(WPA_CIPHER_CCMP | WPA_CIPHER_TKIP | \
                      WPA_CIPHER_WEP104 | WPA_CIPHER_WEP40)
```

15.389 wpa_supplicant/config_winreg.c File Reference

WPA Supplicant / Configuration backend: Windows registry. `#include "includes.h"`

`#include "common.h"`

`#include "uuid.h"`

`#include "config.h"`

Defines

- #define **WPA_KEY_ROOT** HKEY_LOCAL_MACHINE
- #define **WPA_KEY_PREFIX** TEXT("SOFTWARE\\wpa_supplicant")
- #define **TSTR** "%s"
- #define **TNAMELEN** 255
- #define **STR**(t) write_str(netw, #t, ssid)
- #define **INT**(t) write_int(netw, #t, ssid->t, 0)
- #define **INTe**(t) write_int(netw, #t, ssid->eap.t, 0)
- #define **INT_DEF**(t, def) write_int(netw, #t, ssid->t, def)
- #define **INT_DEFe**(t, def) write_int(netw, #t, ssid->eap.t, def)

Functions

- struct 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.

15.389.1 Detailed Description

WPA Supplicant / Configuration backend: Windows registry.

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

This file implements a configuration backend for Windows registry. All the configuration information is stored in the registry and the format for network configuration fields is same as described in the sample configuration file, wpa_supplicant.conf.

Configuration data is in *HKEY_LOCAL_MACHINE\SOFTWARE\wpa_supplicant\configs* key. Each configuration profile has its own key under this. In terms of text files, each profile would map to a separate text file with possibly multiple networks. Under each profile, there is a networks key that lists all networks

as a subkey. Each network has set of values in the same way as network block in the configuration file. In addition, blobs subkey has possible blobs as values.

Example network configuration block:

```
HKEY_LOCAL_MACHINE\SOFTWARE\wpa_supplicant\configs\test\networks\0000
   ssid="example"
   key_mgmt=WPA-PSK
```

15.389.2 Function Documentation

15.389.2.1 struct wpa_config* wpa_config_read (const char * *name*) [read]

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.

15.389.2.2 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.

15.390 wpa_supplicant/ctrl_iface_dbus.c File Reference

WPA Supplicant / dbus-based control interface. `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "config.h"
#include "wpa_supplicant_i.h"
#include "drivers/driver.h"
#include "wps/wps.h"
#include "ctrl_iface_dbus.h"
#include "ctrl_iface_dbus_handlers.h"
```

Data Structures

- struct ctrl_iface_dbus_priv

Defines

- #define **_DBUS_VERSION** (DBUS_VERSION_MAJOR << 8 | DBUS_VERSION_MINOR)
- #define **DBUS_VER**(major, minor) ((major) << 8 | (minor))
- #define **dbus_watch_get_unix_fd** dbus_watch_get_fd

Functions

- u32 wpa_supplicant_dbus_next_objid (struct ctrl_iface_dbus_priv *iface)

  Return next available object id.

- char * wpas_dbus_decompose_object_path (const char *path, char **network, char **bssid)

  Decompose an interface object path into parts.

- DBusMessage * wpas_dbus_new_invalid_iface_error (DBusMessage *message)

  Return a new invalid interface error message.

- DBusMessage * wpas_dbus_new_invalid_network_error (DBusMessage *message)

  Return a new invalid network error message.

- void wpa_supplicant_dbus_notify_scan_results (struct wpa_supplicant *wpa_s)

  Send a scan results signal.

- void wpa_supplicant_dbus_notify_state_change (struct wpa_supplicant *wpa_s, wpa_states new_state, wpa_states old_state)

  Send a state change signal.

- void wpa_supplicant_dbus_notify_scanning (struct wpa_supplicant *wpa_s)

  send scanning status

- void **wpa_supplicant_dbus_notify_wps_cred** (struct wpa_supplicant *wpa_s, const struct wps_credential *cred)
- struct ctrl_iface_dbus_priv * wpa_supplicant_dbus_ctrl_iface_init (struct wpa_global *global)

  Initialize dbus control interface.

- void wpa_supplicant_dbus_ctrl_iface_deinit (struct ctrl_iface_dbus_priv *iface)

  Deinitialize dbus ctrl interface.

- int wpas_dbus_register_iface (struct wpa_supplicant *wpa_s)

  Register a new interface with dbus.

- int wpas_dbus_unregister_iface (struct wpa_supplicant *wpa_s)

  Unregister an interface from dbus.

- struct wpa_supplicant * wpa_supplicant_get_iface_by_dbus_path (struct wpa_global *global, const char *path)

  Get a new network interface.

- int wpa_supplicant_set_dbus_path (struct wpa_supplicant *wpa_s, const char *path)

  Assign a dbus path to an interface.

- const char * wpa_supplicant_get_dbus_path (struct wpa_supplicant *wpa_s)

  Get an interface's dbus path.

15.390.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.390.2 Function Documentation

15.390.2.1 void wpa_supplicant_dbus_ctrl_iface_deinit (struct ctrl_iface_dbus_priv * *iface*)

Deinitialize dbus ctrl interface.

Parameters:

iface Pointer to dbus private data from wpa_supplicant_dbus_ctrl_iface_init()

Deinitialize the dbus control interface that was initialized with wpa_supplicant_dbus_ctrl_iface_init().

15.390.2.2 struct ctrl_iface_dbus_priv* wpa_supplicant_dbus_ctrl_iface_init (struct wpa_global * *global*) [read]

Initialize dbus control interface.

Parameters:

global Pointer to global data from wpa_supplicant_init()

Returns:

Pointer to dbus_ctrl_iface date or NULL on failure

Initialize the dbus control interface and start receiving commands from external programs over the bus.

15.390.2.3 u32 wpa_supplicant_dbus_next_objid (struct ctrl_iface_dbus_priv * *iface*)

Return next available object id.

Parameters:

iface dbus control interface private data

Returns:

Object id

15.390.2.4 void wpa_supplicant_dbus_notify_scan_results (struct wpa_supplicant * *wpa_s*)

Send a scan results signal.

Parameters:

wpa_s wpa_supplicant network interface data

Returns:

0 on success, -1 on failure

Notify listeners that this interface has updated scan results.

15.390.2.5 void wpa_supplicant_dbus_notify_scanning (struct wpa_supplicant * *wpa_s*)

send scanning status

Parameters:

wpa_s wpa_supplicant network interface data

Returns:

0 on success, -1 on failure

Notify listeners of interface scanning state changes

15.390.2.6 void wpa_supplicant_dbus_notify_state_change (struct wpa_supplicant * *wpa_s*, wpa_states *new_state*, wpa_states *old_state*)

Send a state change signal.

Parameters:

wpa_s wpa_supplicant network interface data

new_state new state wpa_supplicant is entering

old_state old state wpa_supplicant is leaving

Returns:

0 on success, -1 on failure

Notify listeners that wpa_supplicant has changed state

15.390.2.7 const char* wpa_supplicant_get_dbus_path (struct wpa_supplicant * *wpa_s*)

Get an interface's dbus path.

Parameters:

wpa_s wpa_supplicant interface structure

Returns:

Interface's dbus object path, or NULL on error

15.390.2.8 struct wpa_supplicant* wpa_supplicant_get_iface_by_dbus_path (struct wpa_global * *global*, const char * *path*) [read]

Get a new network interface.

Parameters:

global Pointer to global data from wpa_supplicant_init()

path Pointer to a dbus object path representing an interface

Returns:

Pointer to the interface or NULL if not found

15.390.2.9 int wpa_supplicant_set_dbus_path (struct wpa_supplicant * *wpa_s*, const char * *path*)

Assign a dbus path to an interface.

Parameters:

wpa_s wpa_supplicant interface structure

path dbus path to set on the interface

Returns:

0 on succes, -1 on error

15.390.2.10 char* wpas_dbus_decompose_object_path (const char * *path*, char ** *network*, char ** *bssid*)

Decompose an interface object path into parts.

Parameters:

path The dbus object path

network (out) the configured network this object path refers to, if any

bssid (out) the scanned bssid this object path refers to, if any

Returns:

The object path of the network interface this path refers to

For a given object path, decomposes the object path into object id, network, and BSSID parts, if those parts exist.

15.390.2.11 DBusMessage* wpas_dbus_new_invalid_iface_error (DBusMessage * *message*)

Return a new invalid interface error message.

Parameters:

message Pointer to incoming dbus message this error refers to

Returns:

A dbus error message

Convenience function to create and return an invalid interface error

15.390.2.12 DBusMessage* wpas_dbus_new_invalid_network_error (DBusMessage * *message*)

Return a new invalid network error message.

Parameters:

message Pointer to incoming dbus message this error refers to

Returns:

a dbus error message

Convenience function to create and return an invalid network error

15.390.2.13 int wpas_dbus_register_iface (struct wpa_supplicant * *wpa_s*)

Register a new interface with dbus.

Parameters:

wpa_s wpa_supplicant interface description structure to register

Returns:

0 on success, -1 on error

Registers a new interface with dbus and assigns it a dbus object path.

15.390.2.14 int wpas_dbus_unregister_iface (struct wpa_supplicant * *wpa_s*)

Unregister an interface from dbus.

Parameters:

wpa_s wpa_supplicant interface structure

Returns:

0 on success, -1 on failure

Unregisters the interface with dbus

15.391 wpa_supplicant/ctrl_iface_dbus.h File Reference

WPA Supplicant / dbus-based control interface.

15.391.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

15.392 wpa_supplicant/ctrl_iface_dbus_handlers.c File Reference

WPA Supplicant / dbus-based control interface. `#include "includes.h"`

```
#include "common.h"
#include "config.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "ctrl_iface_dbus.h"
#include "ctrl_iface_dbus_handlers.h"
#include "notify.h"
#include "eap_peer/eap_methods.h"
#include "dbus_dict_helpers.h"
#include "ieee802_11_defs.h"
#include "wpas_glue.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "wps_supplicant.h"
#include "wpa.h"
```

Functions

- DBusMessage * wpas_dbus_global_add_interface (DBusMessage *message, struct wpa_global *global)

  Request registration of a network interface.

- DBusMessage * wpas_dbus_global_remove_interface (DBusMessage *message, struct wpa_global *global)

  Request deregistration of an interface.

- DBusMessage * wpas_dbus_global_get_interface (DBusMessage *message, struct wpa_global *global)

  Get the object path for an interface name.

- DBusMessage * wpas_dbus_global_set_debugparams (DBusMessage *message, struct wpa_global *global)
- DBusMessage * wpas_dbus_iface_scan (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Request a wireless scan on an interface.

- DBusMessage * wpas_dbus_iface_scan_results (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get the results of a recent scan request.

- DBusMessage * wpas_dbus_bssid_properties (DBusMessage *message, struct wpa_supplicant *wpa_s, struct wpa_scan_res *res)

  Return the properties of a scanned network.

- DBusMessage * wpas_dbus_iface_capabilities (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Return interface capabilities.

- DBusMessage * wpas_dbus_iface_add_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Add a new configured network.

- DBusMessage * wpas_dbus_iface_remove_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Remove a configured network.

- DBusMessage * wpas_dbus_iface_set_network (DBusMessage *message, struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  Set options for a configured network.

- DBusMessage * wpas_dbus_iface_enable_network (DBusMessage *message, struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  Mark a configured network as enabled.

- DBusMessage * wpas_dbus_iface_disable_network (DBusMessage *message, struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  Mark a configured network as disabled.

- DBusMessage * wpas_dbus_iface_select_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Attempt association with a configured network.

- DBusMessage * wpas_dbus_iface_disconnect (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Terminate the current connection.

- DBusMessage * wpas_dbus_iface_set_ap_scan (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Control roaming mode.

- DBusMessage * wpas_dbus_iface_set_smartcard_modules (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Set smartcard related module paths.

- DBusMessage * wpas_dbus_iface_get_state (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface state.

- DBusMessage * wpas_dbus_iface_get_scanning (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface scanning state.

- DBusMessage * wpas_dbus_iface_set_blobs (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Store named binary blobs (ie, for certificates).

- DBusMessage * wpas_dbus_iface_remove_blobs (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Remove named binary blobs.

Variables

- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- int **wpa_debug_timestamp**

15.392.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.392.2 Function Documentation

15.392.2.1 DBusMessage* wpas_dbus_bssid_properties (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*, struct wpa_scan_res * *res*)

Return the properties of a scanned network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

res wpa_supplicant scan result for which to get properties

Returns:

a dbus message containing the properties for the requested network

Handler function for "properties" method call of a scanned network. Returns a dbus message containing the the properties.

15.392.2.2 DBusMessage* wpas_dbus_global_add_interface (DBusMessage * *message*, struct wpa_global * *global*)

Request registration of a network interface.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

The object path of the new interface object, or a dbus error message with more information

Handler function for "addInterface" method call. Handles requests by dbus clients to register a network interface that wpa_supplicant will manage.

15.392.2.3 DBusMessage* wpas_dbus_global_get_interface (DBusMessage * *message*, struct wpa_global * *global*)

Get the object path for an interface name.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

The object path of the interface object, or a dbus error message with more information

Handler function for "getInterface" method call. Handles requests by dbus clients for the object path of an specific network interface.

15.392.2.4 DBusMessage* wpas_dbus_global_remove_interface (DBusMessage * *message*, struct wpa_global * *global*)

Request deregistration of an interface.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

a dbus message containing a UINT32 indicating success (1) or failure (0), or returns a dbus error message with more information

Handler function for "removeInterface" method call. Handles requests by dbus clients to deregister a network interface that wpa_supplicant currently manages.

15.392.2.5 DBusMessage* wpas_dbus_global_set_debugparams (DBusMessage * *message*, struct wpa_global * *global*)

wpas_dbus_global_set_debugparams- Set the debug params : Pointer to incoming dbus message : wpa_-supplicant global data structure Returns: a dbus message containing a UINT32 indicating success (1) or failure (0), or returns a dbus error message with more information

Handler function for "setDebugParams" method call. Handles requests by dbus clients for the object path of an specific network interface.

15.392.2.6 DBusMessage* wpas_dbus_iface_add_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Add a new configured network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing the object path of the new network

Handler function for "addNetwork" method call of a network interface.

15.392.2.7 DBusMessage* wpas_dbus_iface_capabilities (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Return interface capabilities.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a dict of strings

Handler function for "capabilities" method call of an interface.

15.392.2.8 DBusMessage* wpas_dbus_iface_disable_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Mark a configured network as disabled.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

ssid wpa_ssid structure for a configured network

Returns:

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "disable" method call of a configured network.

15.392.2.9 DBusMessage* wpas_dbus_iface_disconnect (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Terminate the current connection.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "disconnect" method call of network interface.

15.392.2.10 DBusMessage* wpas_dbus_iface_enable_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Mark a configured network as enabled.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

ssid wpa_ssid structure for a configured network

Returns:

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "enable" method call of a configured network.

15.392.2.11 DBusMessage* wpas_dbus_iface_get_scanning (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface scanning state.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing whether the interface is scanning

Handler function for "scanning" method call.

15.392.2.12 DBusMessage* wpas_dbus_iface_get_state (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface state.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a STRING representing the current interface state

Handler function for "state" method call.

15.392.2.13 DBusMessage* wpas_dbus_iface_remove_blobs (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Remove named binary blobs.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant data structure

Returns:

A dbus message containing a UINT32 indicating success (1) or failure (0)

Asks wpa_supplicant to remove one or more previously stored binary blobs.

15.392.2.14 DBusMessage* wpas_dbus_iface_remove_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Remove a configured network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "removeNetwork" method call of a network interface.

15.392.2.15 DBusMessage* wpas_dbus_iface_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Request a wireless scan on an interface.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

a dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "scan" method call of a network device. Requests that wpa_supplicant perform a wireless scan as soon as possible on a particular wireless interface.

15.392.2.16 DBusMessage* wpas_dbus_iface_scan_results (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get the results of a recent scan request.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

a dbus message containing a dbus array of objects paths, or returns a dbus error message if not scan results could be found

Handler function for "scanResults" method call of a network device. Returns a dbus message containing the object paths of wireless networks found.

15.392.2.17 DBusMessage* wpas_dbus_iface_select_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Attempt association with a configured network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "selectNetwork" method call of network interface.

15.392.2.18 DBusMessage* wpas_dbus_iface_set_ap_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Control roaming mode.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "setAPScan" method call.

15.392.2.19 DBusMessage* wpas_dbus_iface_set_blobs (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Store named binary blobs (ie, for certificates).

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant data structure

Returns:

A dbus message containing a UINT32 indicating success (1) or failure (0)

Asks wpa_supplicant to internally store a one or more binary blobs.

15.392.2.20 DBusMessage* wpas_dbus_iface_set_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Set options for a configured network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

ssid wpa_ssid structure for a configured network

Returns:

a dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "set" method call of a configured network.

15.392.2.21 DBusMessage* wpas_dbus_iface_set_smartcard_modules (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Set smartcard related module paths.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a UINT32 indicating success (1) or failure (0)

Handler function for "setSmartcardModules" method call.

15.393 wpa_supplicant/ctrl_iface_dbus_handlers.h File Reference

WPA Supplicant / dbus-based control interface.

15.393.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.394 wpa_supplicant/ctrl_iface_dbus_new.c File Reference

WPA Supplicant / dbus-based control interface. `#include "includes.h"`

```
#include "common.h"
#include "config.h"
#include "wpa_supplicant_i.h"
#include "drivers/driver.h"
#include "wps/wps.h"
#include "ctrl_iface_dbus_new_helpers.h"
#include "dbus_dict_helpers.h"
#include "ctrl_iface_dbus_new.h"
#include "ctrl_iface_dbus_new_handlers.h"
```

Data Structures

- struct wpas_dbus_method
- struct wpas_dbus_property
- struct wpas_dbus_signal

Functions

- struct wpas_dbus_callbacks * **wpas_dbus_get_callbacks** (void)
- const char * wpas_dbus_get_path (struct wpa_supplicant *wpa_s)

  Get an interface's dbus path.

15.394.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc. Copyright (c) 2009, Witold Sowa <witold.sowa@gmail.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.

15.394.2 Function Documentation

15.394.2.1 const char* wpas_dbus_get_path (struct wpa_supplicant * *wpa_s*)

Get an interface's dbus path.

Parameters:

wpa_s wpa_supplicant interface structure

Returns:

Interface's dbus object path, or NULL on error

15.395 wpa_supplicant/ctrl_iface_dbus_new.h File Reference

WPA Supplicant / dbus-based control interface.

Data Structures

- struct wpas_dbus_callbacks

Enumerations

- enum **wpas_dbus_prop** { **WPAS_DBUS_PROP_AP_SCAN**, **WPAS_DBUS_PROP_SCANNING**, **WPAS_DBUS_PROP_CURRENT_BSS**, **WPAS_DBUS_PROP_CURRENT_NETWORK** }

15.395.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

15.396 wpa_supplicant/ctrl_iface_dbus_new_handlers.c File Reference

WPA Supplicant / dbus-based control interface. `#include "includes.h"`

```
#include "common.h"
#include "config.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "ctrl_iface_dbus_new_helpers.h"
#include "ctrl_iface_dbus_new.h"
#include "ctrl_iface_dbus_new_handlers.h"
#include "notify.h"
#include "eap_peer/eap_methods.h"
#include "dbus_dict_helpers.h"
#include "ieee802_11_defs.h"
#include "wpas_glue.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "wps_supplicant.h"
```

Defines

- #define **FREQS_ALLOC_CHUNK** 32

Functions

- DBusMessage * wpas_dbus_handler_create_interface (DBusMessage *message, struct wpa_global *global)

  Request registration of a network iface.

- DBusMessage * wpas_dbus_handler_remove_interface (DBusMessage *message, struct wpa_global *global)

  Request deregistration of an interface.

- DBusMessage * wpas_dbus_handler_get_interface (DBusMessage *message, struct wpa_global *global)

  Get the object path for an interface name.

- DBusMessage * wpas_dbus_getter_debug_params (DBusMessage *message, struct wpa_global *global)

  Get the debug params.

- DBusMessage * wpas_dbus_setter_debug_params (DBusMessage *message, struct wpa_global *global)

  Set the debug params.

- DBusMessage * wpas_dbus_getter_interfaces (DBusMessage *message, struct wpa_global *global)

  Request registered interfaces list.

- DBusMessage * wpas_dbus_getter_eap_methods (DBusMessage *message, void *nothing)

  Request supported EAP methods list.

- DBusMessage * wpas_dbus_handler_scan (DBusMessage *message, struct wpa_supplicant *wpa_-s)

  Request a wireless scan on an interface.

- DBusMessage * **wpas_dbus_handler_disconnect** (DBusMessage *message, struct wpa_supplicant *wpa_s)
- DBusMessage * wpas_dbus_handler_add_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Add a new configured network.

- DBusMessage * wpas_dbus_handler_remove_network (DBusMessage *message, struct wpa_-supplicant *wpa_s)

  Remove a configured network.

- DBusMessage * wpas_dbus_handler_select_network (DBusMessage *message, struct wpa_-supplicant *wpa_s)

  Attempt association with a network.

- DBusMessage * wpas_dbus_handler_add_blob (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Store named binary blob (ie, for certificates).

- DBusMessage * wpas_dbus_handler_get_blob (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get named binary blob (ie, for certificates).

- DBusMessage * wpas_dbus_handler_remove_blob (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Remove named binary blob.

- DBusMessage * wpas_dbus_getter_capabilities (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Return interface capabilities.

- DBusMessage * wpas_dbus_getter_state (DBusMessage *message, struct wpa_supplicant *wpa_-s)

  Get interface state.

- DBusMessage * wpas_dbus_getter_scanning (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface scanning state.

- DBusMessage * wpas_dbus_getter_ap_scan (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Control roaming mode.

- DBusMessage * wpas_dbus_setter_ap_scan (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Control roaming mode.

- DBusMessage * wpas_dbus_getter_ifname (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface name.

- DBusMessage * wpas_dbus_getter_driver (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface name.

- DBusMessage * wpas_dbus_getter_current_bss (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get current bss object path.

- DBusMessage * wpas_dbus_getter_current_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get current network object path.

- DBusMessage * wpas_dbus_getter_bridge_ifname (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface name.

- DBusMessage * wpas_dbus_getter_bsss (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get array of BSSs objects.

- DBusMessage * wpas_dbus_getter_networks (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get array of networks objects.

- DBusMessage * wpas_dbus_getter_blobs (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get all blobs defined for this interface.

- DBusMessage * wpas_dbus_getter_bss_properties (DBusMessage *message, struct bss_handler_args *bss)

  Return the properties of a scanned bss.

- DBusMessage * wpas_dbus_getter_enabled (DBusMessage *message, struct network_handler_args *net)

  Check whether network is enabled or disabled.

- DBusMessage * wpas_dbus_setter_enabled (DBusMessage *message, struct network_handler_args *net)

  Mark a configured network as enabled or disabled.

- DBusMessage * wpas_dbus_getter_network_properties (DBusMessage *message, struct network_-handler_args *net)

  Get options for a configured network.

- DBusMessage * wpas_dbus_setter_network_properties (DBusMessage *message, struct network_-handler_args *net)

  Set options for a configured network.

Variables

- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- int **wpa_debug_timestamp**

15.396.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc. Copyright (c) 2009, Witold Sowa <witold.sowa@gmail.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.

15.396.2 Function Documentation

15.396.2.1 DBusMessage* wpas_dbus_getter_ap_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Control roaming mode.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A message containong value of ap_scan variable

Getter function for "ApScan" property.

15.396.2.2 DBusMessage* wpas_dbus_getter_blobs (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get all blobs defined for this interface.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

a dbus message containing a dictionary of pairs (blob_name, blob)

Getter for "Blobs" property.

15.396.2.3 DBusMessage* wpas_dbus_getter_bridge_ifname (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface name.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a name of bridge network interface associated with with wpa_s

Getter for "BridgeIfname" property.

15.396.2.4 DBusMessage* wpas_dbus_getter_bss_properties (DBusMessage * *message*, struct bss_handler_args * *bss*)

Return the properties of a scanned bss.

Parameters:

message Pointer to incoming dbus message

bss a pair of interface describing structure and bss' bssid

Returns:

a dbus message containing the properties for the requested bss

Getter for "Properties" property.

15.396.2.5 DBusMessage* wpas_dbus_getter_bsss (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get array of BSSs objects.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

a dbus message containing an array of all known BSS objects dbus paths

Getter for "BSSs" property.

15.396.2.6 DBusMessage* wpas_dbus_getter_capabilities (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Return interface capabilities.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a dict of strings

Getter for "Capabilities" property of an interface.

15.396.2.7 DBusMessage* wpas_dbus_getter_current_bss (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get current bss object path.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a DBus object path to current BSS

Getter for "CurrentBSS" property.

15.396.2.8 DBusMessage* wpas_dbus_getter_current_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get current network object path.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a DBus object path to current network

Getter for "CurrentNetwork" property.

15.396.2.9 DBusMessage* wpas_dbus_getter_debug_params (DBusMessage * *message*, struct wpa_global * *global*)

Get the debug params.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

DBus message with struct containing debug params.

Getter for "DebugParams" property.

15.396.2.10 DBusMessage* wpas_dbus_getter_driver (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface name.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a name of network interface driver associated with with wpa_s

Getter for "Driver" property.

15.396.2.11 DBusMessage* wpas_dbus_getter_eap_methods (DBusMessage * *message*, void * *nothing*)

Request supported EAP methods list.

Parameters:

message Pointer to incoming dbus message

nothing not used argument. may be NULL or anything else

Returns:

The object paths array containing supported EAP methods represented by strings or DBus error on failure

Getter for "EapMethods" property. Handles requests by dbus clients to return list of strings with supported EAP methods

15.396.2.12 DBusMessage* wpas_dbus_getter_enabled (DBusMessage * *message*, struct network_handler_args * *net*)

Check whether network is enabled or disabled.

Parameters:

message Pointer to incoming dbus message

wpas_dbus_setter_enabled wpa_supplicant structure for a network interface and wpa_ssid structure for a configured network

Returns:

DBus message with boolean indicating state of configured network or DBus error on failure

Getter for "enabled" property of a configured network.

15.396.2.13 DBusMessage* wpas_dbus_getter_ifname (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface name.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a name of network interface associated with with wpa_s

Getter for "Ifname" property.

15.396.2.14 DBusMessage* wpas_dbus_getter_interfaces (DBusMessage * *message*, struct wpa_global * *global*)

Request registered interfaces list.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

The object paths array containing registered interfaces objects paths or DBus error on failure

Getter for "Interfaces" property. Handles requests by dbus clients to return list of registered interfaces objects paths

15.396.2.15 DBusMessage* wpas_dbus_getter_network_properties (DBusMessage * *message*, struct network_handler_args * *net*)

Get options for a configured network.

Parameters:

message Pointer to incoming dbus message

net wpa_supplicant structure for a network interface and wpa_ssid structure for a configured network

Returns:

DBus message with network properties or DBus error on failure

Getter for "Properties" property of a configured network.

15.396.2.16 DBusMessage* wpas_dbus_getter_networks (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get array of networks objects.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

a dbus message containing an array of all configured networks dbus object paths.

Getter for "Networks" property.

15.396.2.17 DBusMessage* wpas_dbus_getter_scanning (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface scanning state.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing whether the interface is scanning

Getter for "scanning" property.

15.396.2.18 DBusMessage* wpas_dbus_getter_state (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface state.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a STRING representing the current interface state

Getter for "State" property.

15.396.2.19 DBusMessage* wpas_dbus_handler_add_blob (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Store named binary blob (ie, for certificates).

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant data structure

Returns:

A dbus message containing an error on failure or NULL on success

Asks wpa_supplicant to internally store a binary blobs.

15.396.2.20 DBusMessage* wpas_dbus_handler_add_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Add a new configured network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing the object path of the new network

Handler function for "AddNetwork" method call of a network interface.

15.396.2.21 DBusMessage* wpas_dbus_handler_create_interface (DBusMessage * *message*, struct wpa_global * *global*)

Request registration of a network iface.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

The object path of the new interface object, or a dbus error message with more information

Handler function for "addInterface" method call. Handles requests by dbus clients to register a network interface that wpa_supplicant will manage.

15.396.2.22 DBusMessage* wpas_dbus_handler_get_blob (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get named binary blob (ie, for certificates).

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant data structure

Returns:

A dbus message containing array of bytes (blob)

Gets one wpa_supplicant's binary blobs.

15.396.2.23 DBusMessage* wpas_dbus_handler_get_interface (DBusMessage * *message*, struct wpa_global * *global*)

Get the object path for an interface name.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

The object path of the interface object, or a dbus error message with more information

Handler function for "getInterface" method call.

15.396.2.24 DBusMessage* wpas_dbus_handler_remove_blob (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Remove named binary blob.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant data structure

Returns:

NULL on success or dbus error

Asks wpa_supplicant to internally remove a binary blobs.

15.396.2.25 DBusMessage* wpas_dbus_handler_remove_interface (DBusMessage * *message*, struct wpa_global * *global*)

Request deregistration of an interface.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

a dbus message containing a UINT32 indicating success (1) or failure (0), or returns a dbus error message with more information

Handler function for "removeInterface" method call. Handles requests by dbus clients to deregister a network interface that wpa_supplicant currently manages.

15.396.2.26 DBusMessage* wpas_dbus_handler_remove_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Remove a configured network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

NULL on success or dbus error on failure

Handler function for "RemoveNetwork" method call of a network interface.

15.396.2.27 DBusMessage* wpas_dbus_handler_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Request a wireless scan on an interface.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

NULL indicating success or DBus error message on failure

Handler function for "Scan" method call of a network device. Requests that wpa_supplicant perform a wireless scan as soon as possible on a particular wireless interface.

15.396.2.28 DBusMessage* wpas_dbus_handler_select_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Attempt association with a network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

NULL on success or dbus error on failure

Handler function for "SelectNetwork" method call of network interface.

15.396.2.29 DBusMessage* wpas_dbus_setter_ap_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Control roaming mode.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

NULL

Setter function for "ApScan" property.

15.396.2.30 DBusMessage* wpas_dbus_setter_debug_params (DBusMessage * *message*, struct wpa_global * *global*)

Set the debug params.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

NULL indicating success or a dbus error message with more information

Setter for "DebugParams" property.

15.396.2.31 DBusMessage* wpas_dbus_setter_enabled (DBusMessage * *message*, struct network_handler_args * *net*)

Mark a configured network as enabled or disabled.

Parameters:

message Pointer to incoming dbus message

wpas_dbus_setter_enabled wpa_supplicant structure for a network interface and wpa_ssid structure for a configured network

Returns:

NULL indicating success or DBus error on failure

Setter for "Enabled" property of a configured network.

15.396.2.32 DBusMessage* wpas_dbus_setter_network_properties (DBusMessage * *message*, struct network_handler_args * *net*)

Set options for a configured network.

Parameters:

message Pointer to incoming dbus message

net wpa_supplicant structure for a network interface and wpa_ssid structure for a configured network

Returns:

NULL indicating success or DBus error on failure

Setter for "Properties" property of a configured network.

15.397 wpa_supplicant/ctrl_iface_dbus_new_handlers.h File Reference

WPA Supplicant / dbus-based control interface.

Data Structures

- struct network_handler_args
- struct bss_handler_args

Functions

- DBusMessage * wpas_dbus_handler_create_interface (DBusMessage *message, struct wpa_global *global)

  Request registration of a network iface.

- DBusMessage * wpas_dbus_handler_remove_interface (DBusMessage *message, struct wpa_global *global)

  Request deregistration of an interface.

- DBusMessage * wpas_dbus_handler_get_interface (DBusMessage *message, struct wpa_global *global)

  Get the object path for an interface name.

- DBusMessage * wpas_dbus_getter_debug_params (DBusMessage *message, struct wpa_global *global)

  Get the debug params.

- DBusMessage * wpas_dbus_setter_debug_params (DBusMessage *message, struct wpa_global *global)

  Set the debug params.

- DBusMessage * wpas_dbus_getter_interfaces (DBusMessage *message, struct wpa_global *global)

  Request registered interfaces list.

- DBusMessage * wpas_dbus_getter_eap_methods (DBusMessage *message, void *nothing)

  Request supported EAP methods list.

- DBusMessage * wpas_dbus_handler_scan (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Request a wireless scan on an interface.

- DBusMessage * **wpas_dbus_handler_disconnect** (DBusMessage *message, struct wpa_supplicant *wpa_s)
- DBusMessage * wpas_dbus_handler_add_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Add a new configured network.

- DBusMessage * wpas_dbus_handler_remove_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Remove a configured network.

- DBusMessage * wpas_dbus_handler_select_network (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Attempt association with a network.

- DBusMessage * wpas_dbus_handler_add_blob (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Store named binary blob (ie, for certificates).

- DBusMessage * wpas_dbus_handler_get_blob (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get named binary blob (ie, for certificates).

- DBusMessage * wpas_dbus_handler_remove_blob (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Remove named binary blob.

- DBusMessage * wpas_dbus_getter_capabilities (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Return interface capabilities.

- DBusMessage * wpas_dbus_getter_state (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface state.

- DBusMessage * wpas_dbus_getter_scanning (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface scanning state.

- DBusMessage * wpas_dbus_getter_ap_scan (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Control roaming mode.

- DBusMessage * wpas_dbus_setter_ap_scan (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Control roaming mode.

- DBusMessage * wpas_dbus_getter_ifname (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface name.

- DBusMessage * wpas_dbus_getter_driver (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface name.

- DBusMessage * wpas_dbus_getter_bridge_ifname (DBusMessage *message, struct wpa_supplicant *wpa_s)

  Get interface name.

- DBusMessage * wpas_dbus_getter_current_bss (DBusMessage *message, struct wpa_supplicant *wpa_s)
  Get current bss object path.
- DBusMessage * wpas_dbus_getter_current_network (DBusMessage *message, struct wpa_supplicant *wpa_s)
  Get current network object path.
- DBusMessage * wpas_dbus_getter_bsss (DBusMessage *message, struct wpa_supplicant *wpa_s)
  Get array of BSSs objects.
- DBusMessage * wpas_dbus_getter_networks (DBusMessage *message, struct wpa_supplicant *wpa_s)
  Get array of networks objects.
- DBusMessage * wpas_dbus_getter_blobs (DBusMessage *message, struct wpa_supplicant *bss)
  Get all blobs defined for this interface.
- DBusMessage * wpas_dbus_getter_bss_properties (DBusMessage *message, struct bss_handler_args *bss)
  Return the properties of a scanned bss.
- DBusMessage * wpas_dbus_getter_enabled (DBusMessage *message, struct network_handler_args *net)
  Check whether network is enabled or disabled.
- DBusMessage * wpas_dbus_setter_enabled (DBusMessage *message, struct network_handler_args *net)
  Mark a configured network as enabled or disabled.
- DBusMessage * wpas_dbus_getter_network_properties (DBusMessage *message, struct network_handler_args *net)
  Get options for a configured network.
- DBusMessage * wpas_dbus_setter_network_properties (DBusMessage *message, struct network_handler_args *net)
  Set options for a configured network.

15.397.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

15.397.2 Function Documentation

15.397.2.1 DBusMessage* wpas_dbus_getter_ap_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Control roaming mode.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A message containong value of ap_scan variable

Getter function for "ApScan" property.

15.397.2.2 DBusMessage* wpas_dbus_getter_blobs (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get all blobs defined for this interface.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

a dbus message containing a dictionary of pairs (blob_name, blob)

Getter for "Blobs" property.

15.397.2.3 DBusMessage* wpas_dbus_getter_bridge_ifname (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface name.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a name of bridge network interface associated with with wpa_s

Getter for "BridgeIfname" property.

15.397.2.4 DBusMessage* wpas_dbus_getter_bss_properties (DBusMessage * *message*, struct bss_handler_args * *bss*)

Return the properties of a scanned bss.

Parameters:

message Pointer to incoming dbus message

bss a pair of interface describing structure and bss' bssid

Returns:

a dbus message containing the properties for the requested bss

Getter for "Properties" property.

15.397.2.5 DBusMessage* wpas_dbus_getter_bsss (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get array of BSSs objects.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

a dbus message containing an array of all known BSS objects dbus paths

Getter for "BSSs" property.

15.397.2.6 DBusMessage* wpas_dbus_getter_capabilities (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Return interface capabilities.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a dict of strings

Getter for "Capabilities" property of an interface.

15.397.2.7 DBusMessage* wpas_dbus_getter_current_bss (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get current bss object path.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a DBus object path to current BSS

Getter for "CurrentBSS" property.

15.397.2.8 DBusMessage* wpas_dbus_getter_current_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get current network object path.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a DBus object path to current network

Getter for "CurrentNetwork" property.

15.397.2.9 DBusMessage* wpas_dbus_getter_debug_params (DBusMessage * *message*, struct wpa_global * *global*)

Get the debug params.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

DBus message with struct containing debug params.

Getter for "DebugParams" property.

15.397.2.10 DBusMessage* wpas_dbus_getter_driver (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface name.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a name of network interface driver associated with with wpa_s

Getter for "Driver" property.

15.397.2.11 DBusMessage* wpas_dbus_getter_eap_methods (DBusMessage * *message*, void * *nothing*)

Request supported EAP methods list.

Parameters:

message Pointer to incoming dbus message

nothing not used argument. may be NULL or anything else

Returns:

The object paths array containing supported EAP methods represented by strings or DBus error on failure

Getter for "EapMethods" property. Handles requests by dbus clients to return list of strings with supported EAP methods

15.397.2.12 DBusMessage* wpas_dbus_getter_enabled (DBusMessage * *message*, struct network_handler_args * *net*)

Check whether network is enabled or disabled.

Parameters:

message Pointer to incoming dbus message

wpas_dbus_setter_enabled wpa_supplicant structure for a network interface and wpa_ssid structure for a configured network

Returns:

DBus message with boolean indicating state of configured network or DBus error on failure

Getter for "enabled" property of a configured network.

15.397.2.13 DBusMessage* wpas_dbus_getter_ifname (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface name.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a name of network interface associated with with wpa_s

Getter for "Ifname" property.

15.397.2.14 DBusMessage* wpas_dbus_getter_interfaces (DBusMessage * *message*, struct wpa_global * *global*)

Request registered interfaces list.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

The object paths array containing registered interfaces objects paths or DBus error on failure

Getter for "Interfaces" property. Handles requests by dbus clients to return list of registered interfaces objects paths

15.397.2.15 DBusMessage* wpas_dbus_getter_network_properties (DBusMessage * *message*, struct network_handler_args * *net*)

Get options for a configured network.

Parameters:

message Pointer to incoming dbus message

net wpa_supplicant structure for a network interface and wpa_ssid structure for a configured network

Returns:

DBus message with network properties or DBus error on failure

Getter for "Properties" property of a configured network.

15.397.2.16 DBusMessage* wpas_dbus_getter_networks (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get array of networks objects.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

a dbus message containing an array of all configured networks dbus object paths.

Getter for "Networks" property.

15.397.2.17 DBusMessage* wpas_dbus_getter_scanning (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface scanning state.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing whether the interface is scanning

Getter for "scanning" property.

15.397.2.18 DBusMessage* wpas_dbus_getter_state (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get interface state.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing a STRING representing the current interface state

Getter for "State" property.

15.397.2.19 DBusMessage* wpas_dbus_handler_add_blob (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Store named binary blob (ie, for certificates).

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant data structure

Returns:

A dbus message containing an error on failure or NULL on success

Asks wpa_supplicant to internally store a binary blobs.

15.397.2.20 DBusMessage* wpas_dbus_handler_add_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Add a new configured network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

A dbus message containing the object path of the new network

Handler function for "AddNetwork" method call of a network interface.

15.397.2.21 DBusMessage* wpas_dbus_handler_create_interface (DBusMessage * *message*, struct wpa_global * *global*)

Request registration of a network iface.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

The object path of the new interface object, or a dbus error message with more information

Handler function for "addInterface" method call. Handles requests by dbus clients to register a network interface that wpa_supplicant will manage.

15.397.2.22 DBusMessage* wpas_dbus_handler_get_blob (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Get named binary blob (ie, for certificates).

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant data structure

Returns:

A dbus message containing array of bytes (blob)

Gets one wpa_supplicant's binary blobs.

15.397.2.23 DBusMessage* wpas_dbus_handler_get_interface (DBusMessage * *message*, struct wpa_global * *global*)

Get the object path for an interface name.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

The object path of the interface object, or a dbus error message with more information

Handler function for "getInterface" method call.

15.397.2.24 DBusMessage* wpas_dbus_handler_remove_blob (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Remove named binary blob.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant data structure

Returns:

NULL on success or dbus error

Asks wpa_supplicant to internally remove a binary blobs.

15.397.2.25 DBusMessage* wpas_dbus_handler_remove_interface (DBusMessage * *message*, struct wpa_global * *global*)

Request deregistration of an interface.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

a dbus message containing a UINT32 indicating success (1) or failure (0), or returns a dbus error message with more information

Handler function for "removeInterface" method call. Handles requests by dbus clients to deregister a network interface that wpa_supplicant currently manages.

15.397.2.26 DBusMessage* wpas_dbus_handler_remove_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Remove a configured network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

NULL on success or dbus error on failure

Handler function for "RemoveNetwork" method call of a network interface.

15.397.2.27 DBusMessage* wpas_dbus_handler_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Request a wireless scan on an interface.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

NULL indicating success or DBus error message on failure

Handler function for "Scan" method call of a network device. Requests that wpa_supplicant perform a wireless scan as soon as possible on a particular wireless interface.

15.397.2.28 DBusMessage* wpas_dbus_handler_select_network (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Attempt association with a network.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

NULL on success or dbus error on failure

Handler function for "SelectNetwork" method call of network interface.

15.397.2.29 DBusMessage* wpas_dbus_setter_ap_scan (DBusMessage * *message*, struct wpa_supplicant * *wpa_s*)

Control roaming mode.

Parameters:

message Pointer to incoming dbus message

wpa_s wpa_supplicant structure for a network interface

Returns:

NULL

Setter function for "ApScan" property.

15.397.2.30 DBusMessage* wpas_dbus_setter_debug_params (DBusMessage * *message*, struct wpa_global * *global*)

Set the debug params.

Parameters:

message Pointer to incoming dbus message

global wpa_supplicant global data structure

Returns:

NULL indicating success or a dbus error message with more information

Setter for "DebugParams" property.

15.397.2.31 DBusMessage* wpas_dbus_setter_enabled (DBusMessage * *message*, struct network_handler_args * *net*)

Mark a configured network as enabled or disabled.

Parameters:

message Pointer to incoming dbus message

wpas_dbus_setter_enabled wpa_supplicant structure for a network interface and wpa_ssid structure for a configured network

Returns:

NULL indicating success or DBus error on failure

Setter for "Enabled" property of a configured network.

15.397.2.32 DBusMessage* wpas_dbus_setter_network_properties (DBusMessage * *message*, struct network_handler_args * *net*)

Set options for a configured network.

Parameters:

message Pointer to incoming dbus message

net wpa_supplicant structure for a network interface and wpa_ssid structure for a configured network

Returns:

NULL indicating success or DBus error on failure

Setter for "Properties" property of a configured network.

15.398 wpa_supplicant/ctrl_iface_dbus_new_helpers.c File Reference

WPA Supplicant / dbus-based control interface. `#include "includes.h"`

`#include "common.h"`

`#include "eloop.h"`

`#include "ctrl_iface_dbus_new_helpers.h"`

Data Structures

- struct wpa_dbus_method_desc

  DBus method description.

- struct wpa_dbus_signal_desc

  DBus signal description.

- struct wpa_dbus_property_desc

  DBus property description.

Defines

- #define **MAX_SIG_LEN** 256
- #define **MESSAGE_UNHANDLED** (DBusMessage ∗) 1

Functions

- u32 wpa_dbus_next_objid (struct ctrl_iface_dbus_new_priv ∗iface)

  Return next available object id.

- void free_dbus_object_desc (struct wpa_dbus_object_desc ∗obj_dsc)

  Frees object description data structure.

- struct ctrl_iface_dbus_new_priv ∗ wpa_dbus_ctrl_iface_init (void ∗application_data, char ∗dbus_-path, char ∗dbus_service, struct wpa_dbus_object_desc ∗obj_desc)

  Initialize dbus control interface.

- void wpa_dbus_ctrl_iface_deinit (struct ctrl_iface_dbus_new_priv ∗iface)

  Deinitialize dbus ctrl interface.

- int wpa_dbus_register_object_per_iface (struct ctrl_iface_dbus_new_priv ∗ctrl_iface, const char ∗path, const char ∗ifname, struct wpa_dbus_object_desc ∗obj_desc)

  Register a new object with dbus.

- int wpa_dbus_unregister_object_per_iface (struct ctrl_iface_dbus_new_priv ∗ctrl_iface, const char ∗path)

  Unregisters DBus object.

- int wpa_dbus_method_register (struct wpa_dbus_object_desc *obj_dsc, const char *dbus_interface, const char *dbus_method, WPADBusMethodHandler method_handler, void *handler_argument, WPADBusArgumentFreeFunction argument_free_func, const struct wpa_dbus_argument args[ ])

  Registers DBus method for given object.

- int wpa_dbus_signal_register (struct wpa_dbus_object_desc *obj_dsc, const char *dbus_interface, const char *dbus_signal, const struct wpa_dbus_argument args[ ])

  Registers DBus signal for given object.

- int wpa_dbus_property_register (struct wpa_dbus_object_desc *obj_dsc, const char *dbus_interface, const char *dbus_property, const char *type, WPADBusPropertyAccessor getter, WPADBusPropertyAccessor setter, void *user_data, WPADBusArgumentFreeFunction user_data_free_func, enum dbus_prop_access _access)

  Registers DBus property for given object.

- void wpa_dbus_signal_property_changed (struct ctrl_iface_dbus_new_priv *iface, WPADBusPropertyAccessor property_getter, void *getter_arg, const char *path, const char *interface_name, const char *property_name)

  Send a property changed signal.

15.398.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc. Copyright (c) 2009, Witold Sowa <witold.sowa@gmail.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.

15.398.2 Function Documentation

15.398.2.1 void free_dbus_object_desc (struct wpa_dbus_object_desc * *obj_dsc*)

Frees object description data structure.

Parameters:

connection DBus connection

obj_dsc Object description to free

Frees each of properties, methods and signals description lists and the object description structure itself.

15.398.2.2 void wpa_dbus_ctrl_iface_deinit (struct ctrl_iface_dbus_new_priv * *iface*)

Deinitialize dbus ctrl interface.

Parameters:

iface Pointer to dbus private data from wpa_dbus_ctrl_iface_init()

Deinitialize the dbus control interface that was initialized with wpa_dbus_ctrl_iface_init().

15.398.2.3 struct ctrl_iface_dbus_new_priv* wpa_dbus_ctrl_iface_init (void * *application_data*, char * *dbus_path*, char * *dbus_service*, struct wpa_dbus_object_desc * *obj_desc*) [read]

Initialize dbus control interface.

Parameters:

application_data Pointer to application specific data structure

dbus_path DBus path to interface object

dbus_service DBus service name to register with

messageHandler a pointer to function which will handle dbus messages coming on interface

Returns:

Pointer to dbus_new_ctrl_iface date or NULL on failure

Initialize the dbus control interface and start receiving commands from external programs over the bus.

15.398.2.4 int wpa_dbus_method_register (struct wpa_dbus_object_desc * *obj_dsc*, const char * *dbus_interface*, const char * *dbus_method*, WPADBusMethodHandler *method_handler*, void * *handler_argument*, WPADBusArgumentFreeFunction *argument_free_func*, const struct wpa_dbus_argument *args*[ ])

Registers DBus method for given object.

Parameters:

obj_dsc Object description for which a method will be registered

dbus_interface DBus interface under which method will be registered

dbus_method a name the method will be registered with

method_handler a function which will be called to handle this method call

handler_argument an additional argument passed to handler function

argument_free_func function used to free handler argument

args method arguments list

Returns:

Zero on success and -1 on failure

Registers DBus method under given name and interface for the object. Method calls will be handled with given handling function and optional argument passed to this function. Handler function is required to return a DBusMessage pointer which will be response to method call. Any method call before being handled must have registered appropriate handler by using this function.

15.398.2.5 u32 wpa_dbus_next_objid (struct ctrl_iface_dbus_new_priv * *iface*)

Return next available object id.

Parameters:

iface dbus control interface private data

Returns:

Object id

15.398.2.6 int wpa_dbus_property_register (struct wpa_dbus_object_desc * *obj_dsc*, const char * *dbus_interface*, const char * *dbus_property*, const char * *type*, WPADBusPropertyAccessor *getter*, WPADBusPropertyAccessor *setter*, void * *user_data*, WPADBusArgumentFreeFunction *user_data_free_func*, enum dbus_prop_access *_access*)

Registers DBus property for given object.

Parameters:

obj_dsc Object description for which a property will be registered

dbus_interface DBus interface under which method will be registered

dbus_property a name the property will be registered with

type a property type signature in form of DBus type description

getter a function called in order to get property value

setter a function called in order to set property value

user_data additional argument passed to setter or getter

user_data_free_func function used to free additional argument

access property access permissions specifier (R, W or RW)

Returns:

Zero on success and -1 on failure

Registers DBus property under given name and interface for the object. Property are set with giver setter function and get with getter. Additional argument is passed to getter or setter. Getter or setter are required to return DBusMessage which is response to Set/Get method calls. Every property must be registered by this function before being used.

15.398.2.7 int wpa_dbus_register_object_per_iface (struct ctrl_iface_dbus_new_priv * *ctrl_iface*, const char * *path*, const char * *ifname*, struct wpa_dbus_object_desc * *obj_desc*)

Register a new object with dbus.

Parameters:

ctrl_iface pointer to dbus private data

path DBus path to object

ifname interface name

obj_desc description of object's methods, signals and properties

Returns:

0 on success, -1 on error

Registers a new interface with dbus and assigns it a dbus object path.

15.398.2.8 void wpa_dbus_signal_property_changed (struct ctrl_iface_dbus_new_priv * *iface*, WPADBusPropertyAccessor *property_getter*, void * *getter_arg*, const char * *path*, const char * *interface_name*, const char * *property_name*)

Send a property changed signal.

Parameters:

iface dbus priv struct

property_getter propperty getter used to fetch new property value

getter_arg argument passed to property getter

path path to object which property has changed

interface_name signal and property interface

property_name name of property which has changed

Notify listeners about changing value of some property. Signal contains property name and its value fetched using given property getter.

15.398.2.9 int wpa_dbus_signal_register (struct wpa_dbus_object_desc * *obj_dsc*, const char * *dbus_interface*, const char * *dbus_signal*, const struct wpa_dbus_argument *args*[ ])

Registers DBus signal for given object.

Parameters:

obj_dsc Object description for which a signal will be registered

dbus_interface DBus interface under which signal will be registered

dbus_signal a name the signal will be registered with

args signal arguments list

Returns:

Zero on success and -1 on failure

Registers DBus signal under given name and interface for the object. Signal registration is NOT required in order to send signals, but not registered signals will not be respected in introspection data therefore it is highly recommended to register every signal before using it.

15.398.2.10 int wpa_dbus_unregister_object_per_iface (struct ctrl_iface_dbus_new_priv * *ctrl_iface*, const char * *path*)

Unregisters DBus object.

Parameters:

ctrl_iface Pointer to dbus private data

path DBus path to object which will be unregistered

Returns:

Zero on success and -1 on failure

Unregisters DBus object given by its path

15.399 wpa_supplicant/ctrl_iface_dbus_new_helpers.h File Reference

WPA Supplicant / dbus-based control interface. `#include <dbus/dbus.h>`

Data Structures

- struct ctrl_iface_dbus_new_priv
- struct wpa_dbus_object_desc
- struct wpa_dbus_argument

Defines

- #define **END_ARGS** { NULL, NULL, ARG_IN }

Typedefs

- typedef DBusMessage *(* **WPADBusMethodHandler**)(DBusMessage *message, void *user_data)
- typedef void(* **WPADBusArgumentFreeFunction**)(void *handler_arg)
- typedef DBusMessage *(* **WPADBusPropertyAccessor**)(DBusMessage *message, void *user_data)

Enumerations

- enum **dbus_prop_access** { **R**, **W**, **RW** }
- enum **dbus_arg_direction** { **ARG_IN**, **ARG_OUT** }

15.399.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

15.400 wpa_supplicant/ctrl_iface_named_pipe.c File Reference

WPA Supplicant / Windows Named Pipe -based control interface. `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "config.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface.h"
#include "wpa_ctrl.h"
#include <sddl.h>
```

Data Structures

- struct wpa_ctrl_dst
  Internal data structure of control interface clients.
- struct ctrl_iface_priv
- struct wpa_global_dst
- struct ctrl_iface_global_priv

Defines

- #define **_WIN32_WINNT** 0x0500
- #define **WPA_SUPPLICANT_NAMED_PIPE** "WpaSupplicant"
- #define **NAMED_PIPE_PREFIX** TEXT("\\\\.\\pipe\\") TEXT(WPA_SUPPLICANT_-NAMED_PIPE)
- #define **REQUEST_BUFSIZE** 256
- #define **REPLY_BUFSIZE** 4096

Functions

- struct ctrl_iface_priv * wpa_supplicant_ctrl_iface_init (struct wpa_supplicant *wpa_s)
  Initialize control interface.
- void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv *priv)
  Deinitialize control interface.
- void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv *priv)
  Wait for ctrl_iface monitor.
- struct ctrl_iface_global_priv * wpa_supplicant_global_ctrl_iface_init (struct wpa_global *global)
  Initialize global control interface.
- void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv *priv)
  Deinitialize global ctrl interface.

15.400.1 Detailed Description

WPA Supplicant / Windows Named Pipe -based control interface.

Copyright

Copyright (c) 2004-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.400.2 Function Documentation

15.400.2.1 void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv * *priv*)

Deinitialize control interface.

Parameters:

priv Pointer to private data from wpa_supplicant_ctrl_iface_init()

Deinitialize the control interface that was initialized with wpa_supplicant_ctrl_iface_init().

Required to be implemented in each control interface backend.

15.400.2.2 struct ctrl_iface_priv* wpa_supplicant_ctrl_iface_init (struct wpa_supplicant * *wpa_s*) [read]

Initialize control interface.

Parameters:

wpa_s Pointer to wpa_supplicant data

Returns:

Pointer to private data on success, NULL on failure

Initialize the control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

15.400.2.3 void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv * *priv*)

Wait for ctrl_iface monitor.

Parameters:

priv Pointer to private data from wpa_supplicant_ctrl_iface_init()

Wait until the first message from an external program using the control interface is received. This function can be used to delay normal startup processing to allow control interface programs to attach with wpa_supplicant before normal operations are started.

Required to be implemented in each control interface backend.

15.400.2.4 void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv * *priv*)

Deinitialize global ctrl interface.

Parameters:

priv Pointer to private data from wpa_supplicant_global_ctrl_iface_init()

Deinitialize the global control interface that was initialized with wpa_supplicant_global_ctrl_iface_init().

Required to be implemented in each control interface backend.

15.400.2.5 struct ctrl_iface_global_priv* wpa_supplicant_global_ctrl_iface_init (struct wpa_global * *global*) [read]

Initialize global control interface.

Parameters:

global Pointer to global data from wpa_supplicant_init()

Returns:

Pointer to private data on success, NULL on failure

Initialize the global control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

15.401 wpa_supplicant/ctrl_iface_udp.c File Reference

WPA Supplicant / UDP socket -based control interface. `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "config.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface.h"
#include "wpa_ctrl.h"
```

Data Structures

- struct wpa_ctrl_dst

  Internal data structure of control interface clients.

- struct ctrl_iface_priv
- struct ctrl_iface_global_priv

Defines

- #define **COOKIE_LEN** 8

Functions

- struct ctrl_iface_priv * wpa_supplicant_ctrl_iface_init (struct wpa_supplicant *wpa_s)

  Initialize control interface.

- void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv *priv)

  Deinitialize control interface.

- void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv *priv)

  Wait for ctrl_iface monitor.

- struct ctrl_iface_global_priv * wpa_supplicant_global_ctrl_iface_init (struct wpa_global *global)

  Initialize global control interface.

- void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv *priv)

  Deinitialize global ctrl interface.

15.401.1 Detailed Description

WPA Supplicant / UDP socket -based control interface.

Copyright

Copyright (c) 2004-2005, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.401.2 Function Documentation

15.401.2.1 void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv * *priv*)

Deinitialize control interface.

Parameters:

priv Pointer to private data from wpa_supplicant_ctrl_iface_init()

Deinitialize the control interface that was initialized with wpa_supplicant_ctrl_iface_init().

Required to be implemented in each control interface backend.

15.401.2.2 struct ctrl_iface_priv* wpa_supplicant_ctrl_iface_init (struct wpa_supplicant * *wpa_s*) [read]

Initialize control interface.

Parameters:

wpa_s Pointer to wpa_supplicant data

Returns:

Pointer to private data on success, NULL on failure

Initialize the control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

15.401.2.3 void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv * *priv*)

Wait for ctrl_iface monitor.

Parameters:

priv Pointer to private data from wpa_supplicant_ctrl_iface_init()

Wait until the first message from an external program using the control interface is received. This function can be used to delay normal startup processing to allow control interface programs to attach with wpa_-supplicant before normal operations are started.

Required to be implemented in each control interface backend.

15.401.2.4 void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv * *priv*)

Deinitialize global ctrl interface.

Parameters:

priv Pointer to private data from wpa_supplicant_global_ctrl_iface_init()

Deinitialize the global control interface that was initialized with wpa_supplicant_global_ctrl_iface_init().

Required to be implemented in each control interface backend.

15.401.2.5 struct ctrl_iface_global_priv* wpa_supplicant_global_ctrl_iface_init (struct wpa_global * *global*) [read]

Initialize global control interface.

Parameters:

global Pointer to global data from wpa_supplicant_init()

Returns:

Pointer to private data on success, NULL on failure

Initialize the global control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

15.402 wpa_supplicant/ctrl_iface_unix.c File Reference

WPA Supplicant / UNIX domain socket -based control interface. `#include "includes.h"`

```
#include <sys/un.h>
#include <sys/stat.h>
#include <grp.h>
#include <stddef.h>
#include "common.h"
#include "eloop.h"
#include "config.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "wpa_supplicant_i.h"
#include "ctrl_iface.h"
```

Data Structures

- struct wpa_ctrl_dst

  Internal data structure of control interface clients.

- struct ctrl_iface_priv
- struct ctrl_iface_global_priv

Functions

- struct ctrl_iface_priv * wpa_supplicant_ctrl_iface_init (struct wpa_supplicant *wpa_s)

  Initialize control interface.

- void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv *priv)

  Deinitialize control interface.

- void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv *priv)

  Wait for ctrl_iface monitor.

- struct ctrl_iface_global_priv * wpa_supplicant_global_ctrl_iface_init (struct wpa_global *global)

  Initialize global control interface.

- void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv *priv)

  Deinitialize global ctrl interface.

15.402.1 Detailed Description

WPA Supplicant / UNIX domain socket -based control interface.

15.402.2 Function Documentation

15.402.2.1 void wpa_supplicant_ctrl_iface_deinit (struct ctrl_iface_priv * *priv*)

Deinitialize control interface.

Parameters:

priv Pointer to private data from wpa_supplicant_ctrl_iface_init()

Deinitialize the control interface that was initialized with wpa_supplicant_ctrl_iface_init().

Required to be implemented in each control interface backend.

15.402.2.2 struct ctrl_iface_priv* wpa_supplicant_ctrl_iface_init (struct wpa_supplicant * *wpa_s*) [read]

Initialize control interface.

Parameters:

wpa_s Pointer to wpa_supplicant data

Returns:

Pointer to private data on success, NULL on failure

Initialize the control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

15.402.2.3 void wpa_supplicant_ctrl_iface_wait (struct ctrl_iface_priv * *priv*)

Wait for ctrl_iface monitor.

Parameters:

priv Pointer to private data from wpa_supplicant_ctrl_iface_init()

Wait until the first message from an external program using the control interface is received. This function can be used to delay normal startup processing to allow control interface programs to attach with wpa_supplicant before normal operations are started.

Required to be implemented in each control interface backend.

15.402.2.4 void wpa_supplicant_global_ctrl_iface_deinit (struct ctrl_iface_global_priv * *priv*)

Deinitialize global ctrl interface.

Parameters:

priv Pointer to private data from wpa_supplicant_global_ctrl_iface_init()

Deinitialize the global control interface that was initialized with wpa_supplicant_global_ctrl_iface_init().

Required to be implemented in each control interface backend.

15.402.2.5 struct ctrl_iface_global_priv* wpa_supplicant_global_ctrl_iface_init (struct wpa_global * *global*) [read]

Initialize global control interface.

Parameters:

global Pointer to global data from wpa_supplicant_init()

Returns:

Pointer to private data on success, NULL on failure

Initialize the global control interface and start receiving commands from external programs.

Required to be implemented in each control interface backend.

15.403 wpa_supplicant/dbus_dict_helpers.c File Reference

WPA Supplicant / dbus-based control interface. `#include "includes.h"`

`#include <dbus/dbus.h>`

`#include "common.h"`

`#include "dbus_dict_helpers.h"`

Defines

- #define **BYTE_ARRAY_CHUNK_SIZE** 34
- #define **BYTE_ARRAY_ITEM_SIZE** (sizeof(char))
- #define **STR_ARRAY_CHUNK_SIZE** 8
- #define **STR_ARRAY_ITEM_SIZE** (sizeof(char *))

Functions

- dbus_bool_t wpa_dbus_dict_open_write (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_close_write (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_append_string (DBusMessageIter *iter_dict, const char *key, const char *value)
- dbus_bool_t wpa_dbus_dict_append_byte (DBusMessageIter *iter_dict, const char *key, const char value)
- dbus_bool_t wpa_dbus_dict_append_bool (DBusMessageIter *iter_dict, const char *key, const dbus_bool_t value)
- dbus_bool_t wpa_dbus_dict_append_int16 (DBusMessageIter *iter_dict, const char *key, const dbus_int16_t value)
- dbus_bool_t wpa_dbus_dict_append_uint16 (DBusMessageIter *iter_dict, const char *key, const dbus_uint16_t value)
- dbus_bool_t wpa_dbus_dict_append_int32 (DBusMessageIter *iter_dict, const char *key, const dbus_int32_t value)
- dbus_bool_t wpa_dbus_dict_append_uint32 (DBusMessageIter *iter_dict, const char *key, const dbus_uint32_t value)
- dbus_bool_t wpa_dbus_dict_append_int64 (DBusMessageIter *iter_dict, const char *key, const dbus_int64_t value)
- dbus_bool_t wpa_dbus_dict_append_uint64 (DBusMessageIter *iter_dict, const char *key, const dbus_uint64_t value)
- dbus_bool_t wpa_dbus_dict_append_double (DBusMessageIter *iter_dict, const char *key, const double value)
- dbus_bool_t wpa_dbus_dict_append_object_path (DBusMessageIter *iter_dict, const char *key, const char *value)
- dbus_bool_t wpa_dbus_dict_append_byte_array (DBusMessageIter *iter_dict, const char *key, const char *value, const dbus_uint32_t value_len)
- dbus_bool_t wpa_dbus_dict_begin_string_array (DBusMessageIter *iter_dict, const char *key, DBusMessageIter *iter_dict_entry, DBusMessageIter *iter_dict_val, DBusMessageIter *iter_array)
- dbus_bool_t wpa_dbus_dict_string_array_add_element (DBusMessageIter *iter_array, const char *elem)
- dbus_bool_t wpa_dbus_dict_end_string_array (DBusMessageIter *iter_dict, DBusMessageIter *iter_dict_entry, DBusMessageIter *iter_dict_val, DBusMessageIter *iter_array)

- dbus_bool_t wpa_dbus_dict_append_string_array (DBusMessageIter *iter_dict, const char *key, const char **items, const dbus_uint32_t num_items)
- dbus_bool_t wpa_dbus_dict_open_read (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_get_entry (DBusMessageIter *iter_dict, struct wpa_dbus_dict_entry *entry)
- dbus_bool_t wpa_dbus_dict_has_dict_entry (DBusMessageIter *iter_dict)
- void wpa_dbus_dict_entry_clear (struct wpa_dbus_dict_entry *entry)

15.403.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.403.2 Function Documentation

15.403.2.1 dbus_bool_t wpa_dbus_dict_append_bool (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_bool_t *value*)

Add a boolean entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The boolean value

Returns:

TRUE on success, FALSE on failure

15.403.2.2 dbus_bool_t wpa_dbus_dict_append_byte (DBusMessageIter * *iter_dict*, const char * *key*, const char *value*)

Add a byte entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The byte value

Returns:

TRUE on success, FALSE on failure

15.403.2.3 dbus_bool_t wpa_dbus_dict_append_byte_array (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*, const dbus_uint32_t *value_len*)

Add a byte array entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The byte array

value_len The length of the byte array, in bytes

Returns:

TRUE on success, FALSE on failure

15.403.2.4 dbus_bool_t wpa_dbus_dict_append_double (DBusMessageIter * *iter_dict*, const char * *key*, const double *value*)

Add a double-precision floating point entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The double-precision floating point value

Returns:

TRUE on success, FALSE on failure

15.403.2.5 dbus_bool_t wpa_dbus_dict_append_int16 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int16_t *value*)

Add a 16-bit signed integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 16-bit signed integer value

Returns:

TRUE on success, FALSE on failure

15.403.2.6 dbus_bool_t wpa_dbus_dict_append_int32 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int32_t *value*)

Add a 32-bit signed integer to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 32-bit signed integer value

Returns:

TRUE on success, FALSE on failure

15.403.2.7 dbus_bool_t wpa_dbus_dict_append_int64 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int64_t *value*)

Add a 64-bit integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 64-bit integer value

Returns:

TRUE on success, FALSE on failure

15.403.2.8 dbus_bool_t wpa_dbus_dict_append_object_path (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*)

Add a DBus object path entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The DBus object path value

Returns:

TRUE on success, FALSE on failure

15.403.2.9 dbus_bool_t wpa_dbus_dict_append_string (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*)

Add a string entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The string value

Returns:

TRUE on success, FALSE on failure

15.403.2.10 dbus_bool_t wpa_dbus_dict_append_string_array (DBusMessageIter * *iter_dict*, const char * *key*, const char ** *items*, const dbus_uint32_t *num_items*)

Convenience function to add an entire string array to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

items The array of strings

num_items The number of strings in the array

Returns:

TRUE on success, FALSE on failure

15.403.2.11 dbus_bool_t wpa_dbus_dict_append_uint16 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint16_t *value*)

Add a 16-bit unsigned integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 16-bit unsigned integer value

Returns:

TRUE on success, FALSE on failure

15.403.2.12 dbus_bool_t wpa_dbus_dict_append_uint32 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint32_t *value*)

Add a 32-bit unsigned integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 32-bit unsigned integer value

Returns:

TRUE on success, FALSE on failure

15.403.2.13 dbus_bool_t wpa_dbus_dict_append_uint64 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint64_t *value*)

Add a 64-bit unsigned integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 64-bit unsigned integer value

Returns:

TRUE on success, FALSE on failure

15.403.2.14 dbus_bool_t wpa_dbus_dict_begin_string_array (DBusMessageIter * *iter_dict*, const char * *key*, DBusMessageIter * *iter_dict_entry*, DBusMessageIter * *iter_dict_val*, DBusMessageIter * *iter_array*)

Begin a string array entry in the dict

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

iter_dict_entry A private DBusMessageIter provided by the caller to be passed to wpa_dbus_dict_end_string_array()

iter_dict_val A private DBusMessageIter provided by the caller to be passed to wpa_dbus_dict_end_string_array()

iter_array On return, the DBusMessageIter to be passed to wpa_dbus_dict_string_array_add_element()

Returns:

TRUE on success, FALSE on failure

15.403.2.15 dbus_bool_t wpa_dbus_dict_close_write (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

End a dict element in a dbus message. Should be paired with a call to wpa_dbus_dict_open_write().

Parameters:

iter valid dbus message iterator, same as passed to wpa_dbus_dict_open_write()

iter_dict a dbus dict iterator returned from wpa_dbus_dict_open_write()

Returns:

TRUE on success, FALSE on failure

15.403.2.16 dbus_bool_t wpa_dbus_dict_end_string_array (DBusMessageIter * *iter_dict*, DBusMessageIter * *iter_dict_entry*, DBusMessageIter * *iter_dict_val*, DBusMessageIter * *iter_array*)

End a string array dict entry

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

iter_dict_entry A private DBusMessageIter returned from wpa_dbus_dict_end_string_array()

iter_dict_val A private DBusMessageIter returned from wpa_dbus_dict_end_string_array()

iter_array A DBusMessageIter returned from wpa_dbus_dict_end_string_array()

Returns:

TRUE on success, FALSE on failure

15.403.2.17 void wpa_dbus_dict_entry_clear (struct wpa_dbus_dict_entry * *entry*)

Free any memory used by the entry object.

Parameters:

entry The entry object

15.403.2.18 dbus_bool_t wpa_dbus_dict_get_entry (DBusMessageIter * *iter_dict*, struct wpa_dbus_dict_entry * *entry*)

Read the current key/value entry from the dict. Entries are dynamically allocated when needed and must be freed after use with the wpa_dbus_dict_entry_clear() function.

The returned entry object will be filled with the type and value of the next entry in the dict, or the type will be DBUS_TYPE_INVALID if an error occurred.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_read()

entry A valid dict entry object into which the dict key and value will be placed

Returns:

TRUE on success, FALSE on failure

15.403.2.19 dbus_bool_t wpa_dbus_dict_has_dict_entry (DBusMessageIter * *iter_dict*)

Return whether or not there are additional dictionary entries.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_read()

Returns:

TRUE if more dict entries exists, FALSE if no more dict entries exist

15.403.2.20 dbus_bool_t wpa_dbus_dict_open_read (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

Start reading from a dbus dict.

Parameters:

iter A valid DBusMessageIter pointing to the start of the dict

iter_dict (out) A DBusMessageIter to be passed to wpa_dbus_dict_read_next_entry()

Returns:

TRUE on success, FALSE on failure

15.403.2.21 dbus_bool_t wpa_dbus_dict_open_write (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

Start a dict in a dbus message. Should be paired with a call to wpa_dbus_dict_close_write().

Parameters:

iter A valid dbus message iterator

iter_dict (out) A dict iterator to pass to further dict functions

Returns:

TRUE on success, FALSE on failure

15.403.2.22 dbus_bool_t wpa_dbus_dict_string_array_add_element (DBusMessageIter * *iter_array*, const char * *elem*)

Add a single string element to a string array dict entry

Parameters:

iter_array A valid DBusMessageIter returned from wpa_dbus_dict_begin_string_array()'s iter_array parameter

elem The string element to be added to the dict entry's string array

Returns:

TRUE on success, FALSE on failure

15.404 wpa_supplicant/dbus_dict_helpers.h File Reference

WPA Supplicant / dbus-based control interface.

Data Structures

- struct wpa_dbus_dict_entry

Functions

- dbus_bool_t wpa_dbus_dict_open_write (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_close_write (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_append_string (DBusMessageIter *iter_dict, const char *key, const char *value)
- dbus_bool_t wpa_dbus_dict_append_byte (DBusMessageIter *iter_dict, const char *key, const char value)
- dbus_bool_t wpa_dbus_dict_append_bool (DBusMessageIter *iter_dict, const char *key, const dbus_bool_t value)
- dbus_bool_t wpa_dbus_dict_append_int16 (DBusMessageIter *iter_dict, const char *key, const dbus_int16_t value)
- dbus_bool_t wpa_dbus_dict_append_uint16 (DBusMessageIter *iter_dict, const char *key, const dbus_uint16_t value)
- dbus_bool_t wpa_dbus_dict_append_int32 (DBusMessageIter *iter_dict, const char *key, const dbus_int32_t value)
- dbus_bool_t wpa_dbus_dict_append_uint32 (DBusMessageIter *iter_dict, const char *key, const dbus_uint32_t value)
- dbus_bool_t wpa_dbus_dict_append_int64 (DBusMessageIter *iter_dict, const char *key, const dbus_int64_t value)
- dbus_bool_t wpa_dbus_dict_append_uint64 (DBusMessageIter *iter_dict, const char *key, const dbus_uint64_t value)
- dbus_bool_t wpa_dbus_dict_append_double (DBusMessageIter *iter_dict, const char *key, const double value)
- dbus_bool_t wpa_dbus_dict_append_object_path (DBusMessageIter *iter_dict, const char *key, const char *value)
- dbus_bool_t wpa_dbus_dict_append_byte_array (DBusMessageIter *iter_dict, const char *key, const char *value, const dbus_uint32_t value_len)
- dbus_bool_t wpa_dbus_dict_begin_string_array (DBusMessageIter *iter_dict, const char *key, DBusMessageIter *iter_dict_entry, DBusMessageIter *iter_dict_val, DBusMessageIter *iter_array)
- dbus_bool_t wpa_dbus_dict_string_array_add_element (DBusMessageIter *iter_array, const char *elem)
- dbus_bool_t wpa_dbus_dict_end_string_array (DBusMessageIter *iter_dict, DBusMessageIter *iter_dict_entry, DBusMessageIter *iter_dict_val, DBusMessageIter *iter_array)
- dbus_bool_t wpa_dbus_dict_append_string_array (DBusMessageIter *iter_dict, const char *key, const char **items, const dbus_uint32_t num_items)
- dbus_bool_t wpa_dbus_dict_open_read (DBusMessageIter *iter, DBusMessageIter *iter_dict)
- dbus_bool_t wpa_dbus_dict_get_entry (DBusMessageIter *iter_dict, struct wpa_dbus_dict_entry *entry)
- dbus_bool_t wpa_dbus_dict_has_dict_entry (DBusMessageIter *iter_dict)
- void wpa_dbus_dict_entry_clear (struct wpa_dbus_dict_entry *entry)

15.404.1 Detailed Description

WPA Supplicant / dbus-based control interface.

Copyright

Copyright (c) 2006, Dan Williams <dcbw@redhat.com> and Red Hat, Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.404.2 Function Documentation

15.404.2.1 dbus_bool_t wpa_dbus_dict_append_bool (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_bool_t *value*)

Add a boolean entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The boolean value

Returns:

TRUE on success, FALSE on failure

15.404.2.2 dbus_bool_t wpa_dbus_dict_append_byte (DBusMessageIter * *iter_dict*, const char * *key*, const char *value*)

Add a byte entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The byte value

Returns:

TRUE on success, FALSE on failure

15.404.2.3 dbus_bool_t wpa_dbus_dict_append_byte_array (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*, const dbus_uint32_t *value_len*)

Add a byte array entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The byte array

value_len The length of the byte array, in bytes

Returns:

TRUE on success, FALSE on failure

15.404.2.4 dbus_bool_t wpa_dbus_dict_append_double (DBusMessageIter * *iter_dict*, const char * *key*, const double *value*)

Add a double-precision floating point entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The double-precision floating point value

Returns:

TRUE on success, FALSE on failure

15.404.2.5 dbus_bool_t wpa_dbus_dict_append_int16 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int16_t *value*)

Add a 16-bit signed integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 16-bit signed integer value

Returns:

TRUE on success, FALSE on failure

15.404.2.6 dbus_bool_t wpa_dbus_dict_append_int32 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int32_t *value*)

Add a 32-bit signed integer to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 32-bit signed integer value

Returns:

TRUE on success, FALSE on failure

15.404.2.7 dbus_bool_t wpa_dbus_dict_append_int64 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_int64_t *value*)

Add a 64-bit integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 64-bit integer value

Returns:

TRUE on success, FALSE on failure

15.404.2.8 dbus_bool_t wpa_dbus_dict_append_object_path (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*)

Add a DBus object path entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The DBus object path value

Returns:

TRUE on success, FALSE on failure

15.404.2.9 dbus_bool_t wpa_dbus_dict_append_string (DBusMessageIter * *iter_dict*, const char * *key*, const char * *value*)

Add a string entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The string value

Returns:

TRUE on success, FALSE on failure

15.404.2.10 dbus_bool_t wpa_dbus_dict_append_string_array (DBusMessageIter * *iter_dict*, const char * *key*, const char ** *items*, const dbus_uint32_t *num_items*)

Convenience function to add an entire string array to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

items The array of strings

num_items The number of strings in the array

Returns:

TRUE on success, FALSE on failure

15.404.2.11 dbus_bool_t wpa_dbus_dict_append_uint16 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint16_t *value*)

Add a 16-bit unsigned integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 16-bit unsigned integer value

Returns:

TRUE on success, FALSE on failure

15.404.2.12 dbus_bool_t wpa_dbus_dict_append_uint32 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint32_t *value*)

Add a 32-bit unsigned integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 32-bit unsigned integer value

Returns:

TRUE on success, FALSE on failure

15.404.2.13 dbus_bool_t wpa_dbus_dict_append_uint64 (DBusMessageIter * *iter_dict*, const char * *key*, const dbus_uint64_t *value*)

Add a 64-bit unsigned integer entry to the dict.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

value The 64-bit unsigned integer value

Returns:

TRUE on success, FALSE on failure

15.404.2.14 dbus_bool_t wpa_dbus_dict_begin_string_array (DBusMessageIter * *iter_dict*, const char * *key*, DBusMessageIter * *iter_dict_entry*, DBusMessageIter * *iter_dict_val*, DBusMessageIter * *iter_array*)

Begin a string array entry in the dict

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

key The key of the dict item

iter_dict_entry A private DBusMessageIter provided by the caller to be passed to wpa_dbus_dict_end_string_array()

iter_dict_val A private DBusMessageIter provided by the caller to be passed to wpa_dbus_dict_end_string_array()

iter_array On return, the DBusMessageIter to be passed to wpa_dbus_dict_string_array_add_element()

Returns:

TRUE on success, FALSE on failure

15.404.2.15 dbus_bool_t wpa_dbus_dict_close_write (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

End a dict element in a dbus message. Should be paired with a call to wpa_dbus_dict_open_write().

Parameters:

iter valid dbus message iterator, same as passed to wpa_dbus_dict_open_write()

iter_dict a dbus dict iterator returned from wpa_dbus_dict_open_write()

Returns:

TRUE on success, FALSE on failure

15.404.2.16 dbus_bool_t wpa_dbus_dict_end_string_array (DBusMessageIter * *iter_dict*, DBusMessageIter * *iter_dict_entry*, DBusMessageIter * *iter_dict_val*, DBusMessageIter * *iter_array*)

End a string array dict entry

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_write()

iter_dict_entry A private DBusMessageIter returned from wpa_dbus_dict_end_string_array()

iter_dict_val A private DBusMessageIter returned from wpa_dbus_dict_end_string_array()

iter_array A DBusMessageIter returned from wpa_dbus_dict_end_string_array()

Returns:

TRUE on success, FALSE on failure

15.404.2.17 void wpa_dbus_dict_entry_clear (struct wpa_dbus_dict_entry * *entry*)

Free any memory used by the entry object.

Parameters:

entry The entry object

15.404.2.18 dbus_bool_t wpa_dbus_dict_get_entry (DBusMessageIter * *iter_dict*, struct wpa_dbus_dict_entry * *entry*)

Read the current key/value entry from the dict. Entries are dynamically allocated when needed and must be freed after use with the wpa_dbus_dict_entry_clear() function.

The returned entry object will be filled with the type and value of the next entry in the dict, or the type will be DBUS_TYPE_INVALID if an error occurred.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_read()

entry A valid dict entry object into which the dict key and value will be placed

Returns:

TRUE on success, FALSE on failure

15.404.2.19 dbus_bool_t wpa_dbus_dict_has_dict_entry (DBusMessageIter * *iter_dict*)

Return whether or not there are additional dictionary entries.

Parameters:

iter_dict A valid DBusMessageIter returned from wpa_dbus_dict_open_read()

Returns:

TRUE if more dict entries exists, FALSE if no more dict entries exist

15.404.2.20 dbus_bool_t wpa_dbus_dict_open_read (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

Start reading from a dbus dict.

Parameters:

iter A valid DBusMessageIter pointing to the start of the dict

iter_dict (out) A DBusMessageIter to be passed to wpa_dbus_dict_read_next_entry()

Returns:

TRUE on success, FALSE on failure

15.404.2.21 dbus_bool_t wpa_dbus_dict_open_write (DBusMessageIter * *iter*, DBusMessageIter * *iter_dict*)

Start a dict in a dbus message. Should be paired with a call to wpa_dbus_dict_close_write().

Parameters:

iter A valid dbus message iterator

iter_dict (out) A dict iterator to pass to further dict functions

Returns:

TRUE on success, FALSE on failure

15.404.2.22 dbus_bool_t wpa_dbus_dict_string_array_add_element (DBusMessageIter * *iter_array*, const char * *elem*)

Add a single string element to a string array dict entry

Parameters:

iter_array A valid DBusMessageIter returned from wpa_dbus_dict_begin_string_array()'s iter_array parameter

elem The string element to be added to the dict entry's string array

Returns:

TRUE on success, FALSE on failure

15.405 wpa_supplicant/eapol_test.c File Reference

WPA Supplicant - test code. `#include "includes.h"`

```
#include <assert.h>
#include "common.h"
#include "config.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "eap_peer/eap.h"
#include "eloop.h"
#include "wpa.h"
#include "eap_peer/eap_i.h"
#include "wpa_supplicant_i.h"
#include "radius/radius.h"
#include "radius/radius_client.h"
#include "ctrl_iface.h"
#include "pcsc_funcs.h"
```

Data Structures

- struct extra_radius_attr
- struct eapol_test_data

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

- int **main** (int argc, char ∗argv[ ])

Variables

- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- struct wpa_driver_ops ∗ **wpa_drivers** [ ] = { NULL }

15.405.1 Detailed Description

WPA Supplicant - test code.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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.

15.406 wpa_supplicant/events.c File Reference

WPA Supplicant - Driver event processing. `#include "includes.h"`

```
#include "common.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "wpa.h"
#include "eloop.h"
#include "config.h"
#include "l2_packet/l2_packet.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "pcsc_funcs.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_ctrl.h"
#include "eap_peer/eap.h"
#include "notify.h"
#include "ieee802_11_defs.h"
#include "blacklist.h"
#include "wpas_glue.h"
#include "wps_supplicant.h"
#include "ibss_rsn.h"
#include "sme.h"
#include "bgscan.h"
```

Functions

- void **wpa_supplicant_mark_disassoc** (struct wpa_supplicant *wpa_s)
- 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 (void *ctx, wpa_event_type event, union wpa_event_data *data)

  Report a driver event for wpa_supplicant.

15.406.1 Detailed Description

WPA Supplicant - Driver event processing.

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.406.2 Function Documentation

15.406.2.1 void wpa_supplicant_event (void * *ctx*, wpa_event_type *event*, union wpa_event_data * *data*)

Report a driver event for wpa_supplicant.

Parameters:

ctx Context pointer (wpa_s); 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.

15.406.2.2 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).

15.407 wpa_supplicant/ibss_rsn.c File Reference

wpa_supplicant - IBSS RSN `#include "includes.h"`

```
#include "common.h"
#include "l2_packet/l2_packet.h"
#include "wpa_supplicant_i.h"
#include "wpa.h"
#include "wpa_ie.h"
#include "../hostapd/wpa.h"
#include "ibss_rsn.h"
```

Functions

- int **ibss_rsn_supp_init** (struct ibss_rsn_peer *peer, const u8 *own_addr, const u8 *psk)
- int **ibss_rsn_start** (struct ibss_rsn *ibss_rsn, const u8 *addr)
- struct ibss_rsn * **ibss_rsn_init** (struct wpa_supplicant *wpa_s)
- void **ibss_rsn_deinit** (struct ibss_rsn *ibss_rsn)
- int **ibss_rsn_rx_eapol** (struct ibss_rsn *ibss_rsn, const u8 *src_addr, const u8 *buf, size_t len)
- void **ibss_rsn_set_psk** (struct ibss_rsn *ibss_rsn, const u8 *psk)

15.407.1 Detailed Description

wpa_supplicant - IBSS RSN

Copyright

15.408 wpa_supplicant/ibss_rsn.h File Reference

wpa_supplicant - IBSS RSN

Data Structures

- struct ibss_rsn_peer
- struct ibss_rsn

Functions

- struct ibss_rsn * **ibss_rsn_init** (struct wpa_supplicant *wpa_s)
- void **ibss_rsn_deinit** (struct ibss_rsn *ibss_rsn)
- int **ibss_rsn_start** (struct ibss_rsn *ibss_rsn, const u8 *addr)
- int **ibss_rsn_rx_eapol** (struct ibss_rsn *ibss_rsn, const u8 *src_addr, const u8 *buf, size_t len)
- void **ibss_rsn_set_psk** (struct ibss_rsn *ibss_rsn, const u8 *psk)

15.408.1 Detailed Description

wpa_supplicant - IBSS RSN

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.409 wpa_supplicant/main_none.c File Reference

WPA Supplicant / Example program entrypoint. `#include "includes.h"`

`#include "common.h"`

`#include "wpa_supplicant_i.h"`

Functions

- int **main** (int argc, char *argv[ ])

15.409.1 Detailed Description

WPA Supplicant / Example program entrypoint.

Copyright

15.410 wpa_supplicant/main_symbian.cpp File Reference

WPA Supplicant / Program entrypoint for Symbian. `#include "includes.h"`

`#include "common.h"`

`#include "wpa_supplicant_i.h"`

Functions

- GLDEF_C TInt **E32Main** (void)

15.410.1 Detailed Description

WPA Supplicant / Program entrypoint for Symbian.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.411 wpa_supplicant/main_winmain.c File Reference

WPA Supplicant / WinMain() function for Windows-based applications. `#include "includes.h"`

`#include "common.h"`

`#include "wpa_supplicant_i.h"`

Defines

- #define **CMDLINE** LPSTR

Functions

- int WINAPI **WinMain** (HINSTANCE hInstance, HINSTANCE hPrevInstance, CMDLINE lpCmdLine, int nShowCmd)

15.411.1 Detailed Description

WPA Supplicant / WinMain() function for Windows-based applications.

Copyright

Copyright (c) 2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.412 wpa_supplicant/main_winsvc.c File Reference

WPA Supplicant / main() function for Win32 service. `#include "includes.h"`

```
#include <windows.h>
#include "common.h"
#include "wpa_supplicant_i.h"
#include "eloop.h"
```

Defines

- #define **WPASVC_NAME** TEXT("wpasvc")
- #define **WPASVC_DISPLAY_NAME** TEXT("wpa_supplicant service")
- #define **WPASVC_DESCRIPTION** TEXT("Provides IEEE 802.1X and WPA/WPA2 supplicant functionality")
- #define **TSTR** "%s"
- #define **TBUFLEN** 255

Functions

- int **main** (int argc, char *argv[ ])

15.412.1 Detailed Description

WPA Supplicant / main() function for Win32 service.

Copyright

Copyright (c) 2003-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

The root of wpa_supplicant configuration in registry is HKEY_LOCAL_MACHINE\SOFTWARE\wpa_supplicant. This level includes global parameters and a 'interfaces' subkey with all the interface configuration (adapter to confname mapping). Each such mapping is a subkey that has 'adapter' and 'config' values.

This program can be run either as a normal command line application, e.g., for debugging, with 'wpasvc.exe app' or as a Windows service. Service need to be registered with 'wpasvc.exe reg <full path to wpasvc.exe>'. After this, it can be started like any other Windows service (e.g., 'net start wpasvc') or it can be configured to start automatically through the Services tool in administrative tasks. The service can be unregistered with 'wpasvc.exe unreg'.

15.413 wpa_supplicant/notify.c File Reference

wpa_supplicant - Event notifications `#include "includes.h"`

```
#include "common.h"
#include "config.h"
#include "wpa_supplicant_i.h"
#include "wps_supplicant.h"
#include "ctrl_iface_dbus.h"
#include "ctrl_iface_dbus_new.h"
#include "notify.h"
```

Functions

- int **wpas_notify_supplicant_initialized** (struct wpa_global *global)
- void **wpas_notify_supplicant_deinitialized** (struct wpa_global *global)
- int **wpas_notify_iface_added** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_iface_removed** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_state_changed** (struct wpa_supplicant *wpa_s, wpa_states new_state, wpa_states old_state)
- void **wpas_notify_network_changed** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_ap_scan_changed** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_bssid_changed** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_network_enabled_changed** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpas_notify_network_selected** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpas_notify_scanning** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_scan_done** (struct wpa_supplicant *wpa_s, int success)
- void **wpas_notify_scan_results** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_wps_credential** (struct wpa_supplicant *wpa_s, const struct wps_credential *cred)
- void **wpas_notify_wps_event_m2d** (struct wpa_supplicant *wpa_s, struct wps_event_m2d *m2d)
- void **wpas_notify_wps_event_fail** (struct wpa_supplicant *wpa_s, struct wps_event_fail *fail)
- void **wpas_notify_wps_event_success** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_network_added** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpas_notify_network_removed** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpas_notify_bss_added** (struct wpa_supplicant *wpa_s, u8 bssid[ ])
- void **wpas_notify_bss_removed** (struct wpa_supplicant *wpa_s, u8 bssid[ ])
- void **wpas_notify_blob_added** (struct wpa_supplicant *wpa_s, const char *name)
- void **wpas_notify_blob_removed** (struct wpa_supplicant *wpa_s, const char *name)
- void **wpas_notify_debug_params_changed** (struct wpa_global *global)

15.413.1 Detailed Description

wpa_supplicant - Event notifications

Copyright

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.414 wpa_supplicant/notify.h File Reference

wpa_supplicant - Event notifications

Functions

- int **wpas_notify_supplicant_initialized** (struct wpa_global *global)
- void **wpas_notify_supplicant_deinitialized** (struct wpa_global *global)
- int **wpas_notify_iface_added** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_iface_removed** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_state_changed** (struct wpa_supplicant *wpa_s, wpa_states new_state, wpa_states old_state)
- void **wpas_notify_network_changed** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_ap_scan_changed** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_bssid_changed** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_network_enabled_changed** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpas_notify_network_selected** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpas_notify_scanning** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_scan_done** (struct wpa_supplicant *wpa_s, int success)
- void **wpas_notify_scan_results** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_wps_credential** (struct wpa_supplicant *wpa_s, const struct wps_credential *cred)
- void **wpas_notify_wps_event_m2d** (struct wpa_supplicant *wpa_s, struct wps_event_m2d *m2d)
- void **wpas_notify_wps_event_fail** (struct wpa_supplicant *wpa_s, struct wps_event_fail *fail)
- void **wpas_notify_wps_event_success** (struct wpa_supplicant *wpa_s)
- void **wpas_notify_network_added** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpas_notify_network_removed** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- void **wpas_notify_bss_added** (struct wpa_supplicant *wpa_s, u8 bssid[ ])
- void **wpas_notify_bss_removed** (struct wpa_supplicant *wpa_s, u8 bssid[ ])
- void **wpas_notify_blob_added** (struct wpa_supplicant *wpa_s, const char *name)
- void **wpas_notify_blob_removed** (struct wpa_supplicant *wpa_s, const char *name)
- void **wpas_notify_debug_params_changed** (struct wpa_global *global)

15.414.1 Detailed Description

wpa_supplicant - Event notifications

Copyright

15.415 wpa_supplicant/preauth_test.c File Reference

WPA Supplicant - test code for pre-authentication. `#include "includes.h"`

```
#include <assert.h>
#include "common.h"
#include "config.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "eloop.h"
#include "wpa.h"
#include "eap_peer/eap.h"
#include "wpa_supplicant_i.h"
#include "l2_packet/l2_packet.h"
#include "ctrl_iface.h"
#include "pcsc_funcs.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "drivers/driver.h"
```

Data Structures

- struct preauth_test_data

Functions

- int **main** (int argc, char *argv[ ])

Variables

- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- struct wpa_driver_ops * **wpa_drivers** [ ] = { NULL }

15.415.1 Detailed Description

WPA Supplicant - test code for pre-authentication.

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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.

15.416 wpa_supplicant/scan.c File Reference

WPA Supplicant - Scanning. `#include "includes.h"`

```
#include "common.h"
#include "eloop.h"
#include "config.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "mlme.h"
#include "wps_supplicant.h"
#include "notify.h"
```

Functions

- int **wpa_supplicant_enabled_networks** (struct wpa_config *conf)
- int **wpa_supplicant_trigger_scan** (struct wpa_supplicant *wpa_s, struct wpa_driver_scan_params *params)
- 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_notify_scanning** (struct wpa_supplicant *wpa_s, int scanning)

15.416.1 Detailed Description

WPA Supplicant - Scanning.

Copyright

Copyright (c) 2003-2008, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.416.2 Function Documentation

15.416.2.1 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().

15.416.2.2 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.

15.417 wpa_supplicant/sme.c File Reference

wpa_supplicant - SME `#include "includes.h"`

```
#include "common.h"
#include "ieee802_11_defs.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "wpa_common.h"
#include "wpa.h"
#include "pmksa_cache.h"
#include "config.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "wpas_glue.h"
#include "wps_supplicant.h"
#include "notify.h"
#include "blacklist.h"
#include "sme.h"
```

Functions

- void **sme_authenticate** (struct wpa_supplicant *wpa_s, struct wpa_scan_res *bss, struct wpa_ssid *ssid)
- void **sme_event_auth** (struct wpa_supplicant *wpa_s, union wpa_event_data *data)
- int **sme_update_ft_ies** (struct wpa_supplicant *wpa_s, const u8 *md, const u8 *ies, size_t ies_len)
- void **sme_event_assoc_reject** (struct wpa_supplicant *wpa_s, union wpa_event_data *data)
- void **sme_event_auth_timed_out** (struct wpa_supplicant *wpa_s, union wpa_event_data *data)
- void **sme_event_assoc_timed_out** (struct wpa_supplicant *wpa_s, union wpa_event_data *data)

15.417.1 Detailed Description

wpa_supplicant - SME

Copyright

Copyright (c) 2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.418 wpa_supplicant/sme.h File Reference

wpa_supplicant - SME

15.418.1 Detailed Description

wpa_supplicant - SME

Copyright

15.419 wpa_supplicant/win_if_list.c File Reference

win_if_list - Display network interfaces with description (for Windows) `#include "includes.h"`

`#include <stdio.h>`

`#include "pcap.h"`

`#include <winsock.h>`

Functions

- int **main** (int argc, char *argv[ ])

15.419.1 Detailed Description

win_if_list - Display network interfaces with description (for Windows)

Copyright

Copyright (c) 2004-2006, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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.

15.420 wpa_supplicant/wpa_cli.c File Reference

WPA Supplicant - command line interface for wpa_supplicant daemon. `#include "includes.h"`

`#include "wpa_ctrl.h"`

`#include "common.h"`

`#include "version.h"`

Data Structures

- struct wpa_cli_cmd

Defines

- #define **max_args** 10

Enumerations

- enum **wpa_cli_cmd_flags** { **cli_cmd_flag_none** = 0x00, **cli_cmd_flag_sensitive** = 0x01 }

Functions

- int **main** (int argc, char *argv[ ])

15.420.1 Detailed Description

WPA Supplicant - command line interface for wpa_supplicant daemon.

Copyright

Copyright (c) 2004-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.421 wpa_supplicant/wpa_passphrase.c File Reference

WPA Supplicant - ASCII passphrase to WPA PSK tool. `#include "includes.h"`

`#include "common.h"`

`#include "sha1.h"`

Functions

- int **main** (int argc, char ∗argv[ ])

15.421.1 Detailed Description

WPA Supplicant - ASCII passphrase to WPA PSK tool.

Copyright

15.422 wpa_supplicant/wpa_priv.c File Reference

WPA Supplicant / privileged helper program. `#include "includes.h"`

```
#include <sys/un.h>
#include <sys/stat.h>
#include "common.h"
#include "eloop.h"
#include "version.h"
#include "drivers/driver.h"
#include "l2_packet/l2_packet.h"
#include "privsep_commands.h"
#include "ieee802_11_defs.h"
```

Data Structures

- struct wpa_priv_interface

Defines

- #define **SCAN_AP_LIMIT** 128

Functions

- void wpa_supplicant_event (void *ctx, wpa_event_type event, union wpa_event_data *data)

  Report a driver event for wpa_supplicant.

- void wpa_supplicant_rx_eapol (void *ctx, const u8 *src_addr, const u8 *buf, size_t len)

  Deliver a received EAPOL frame to wpa_supplicant.

- int **main** (int argc, char *argv[ ])

Variables

- struct wpa_driver_ops * **wpa_drivers** [ ]
- int **wpa_debug_level**

15.422.1 Detailed Description

WPA Supplicant / privileged helper program.

Copyright

Copyright (c) 2007-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.422.2 Function Documentation

15.422.2.1 void wpa_supplicant_event (void * *ctx*, wpa_event_type *event*, union wpa_event_data * *data*)

Report a driver event for wpa_supplicant.

Parameters:

ctx Context pointer (wpa_s); 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.

15.422.2.2 void wpa_supplicant_rx_eapol (void * *ctx*, const u8 * *src_addr*, const u8 * *buf*, size_t *len*)

Deliver a received EAPOL frame to wpa_supplicant.

Parameters:

ctx Context pointer (wpa_s); this is the ctx variable registered with struct wpa_driver_ops::init()

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. Most driver interfaces rely on more generic OS mechanism for receiving frames through l2_packet, but if such a mechanism is not available, the driver wrapper may take care of received EAPOL frames and deliver them to the core supplicant code by calling this function.

15.423 wpa_supplicant/wpa_supplicant.c File Reference

WPA Supplicant. `#include "includes.h"`

```
#include "common.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "eap_peer/eap.h"
#include "eap_server/eap_methods.h"
#include "wpa.h"
#include "eloop.h"
#include "config.h"
#include "l2_packet/l2_packet.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "ctrl_iface.h"
#include "pcsc_funcs.h"
#include "version.h"
#include "preauth.h"
#include "pmksa_cache.h"
#include "wpa_ctrl.h"
#include "mlme.h"
#include "ieee802_11_defs.h"
#include "blacklist.h"
#include "wpas_glue.h"
#include "wps_supplicant.h"
#include "ibss_rsn.h"
#include "sme.h"
#include "ap.h"
#include "notify.h"
#include "bgscan.h"
```

Functions

- int **wpa_set_wep_keys** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- 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.

- 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_res *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_res *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.

- void wpa_supplicant_enable_network (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  Mark a configured network as enabled.

- void wpa_supplicant_disable_network (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  Mark a configured network as disabled.

- void wpa_supplicant_select_network (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  Attempt association with a network.

- int wpa_supplicant_set_ap_scan (struct wpa_supplicant *wpa_s, int ap_scan)

  Set AP scan mode for interface.

- int wpa_supplicant_set_debug_params (struct wpa_global *global, int debug_level, int debug_timestamp, int debug_show_keys)

  Set global debug params.

- int wpa_supplicant_get_scan_results (struct wpa_supplicant *wpa_s)

  Get scan results.

- struct 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.

- void **wpa_supplicant_sta_rx** (void *ctx, const u8 *buf, size_t len, struct ieee80211_rx_status *rx_-status)
- int wpa_supplicant_driver_init (struct wpa_supplicant *wpa_s)

  Initialize driver interface parameters.

- struct 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.

- struct wpa_supplicant * wpa_supplicant_get_iface (struct wpa_global *global, const char *ifname)

  Get a new network interface.

- struct wpa_global * wpa_supplicant_init (struct wpa_params *params)

  Initialize wpa_supplicant.

- int wpa_supplicant_run (struct wpa_global *global)

  Run the wpa_supplicant main event loop.

- void wpa_supplicant_deinit (struct wpa_global *global)

  Deinitialize wpa_supplicant.

Variables

- const char * **wpa_supplicant_version**
- const char * **wpa_supplicant_license**
- const char * **wpa_supplicant_full_license1**
- const char * **wpa_supplicant_full_license2**
- const char * **wpa_supplicant_full_license3**
- const char * **wpa_supplicant_full_license4**
- const char * **wpa_supplicant_full_license5**
- int **wpa_debug_level**
- int **wpa_debug_show_keys**
- int **wpa_debug_timestamp**
- struct wpa_driver_ops * **wpa_drivers** [ ]

15.423.1 Detailed Description

WPA Supplicant.

Copyright

Copyright (c) 2003-2009, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published 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.

15.423.2 Function Documentation

15.423.2.1 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.

15.423.2.2 struct wpa_supplicant* wpa_supplicant_add_iface (struct wpa_global * *global*, struct wpa_interface * *iface*) [read]

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.

15.423.2.3 void wpa_supplicant_associate (struct wpa_supplicant * *wpa_s*, struct wpa_scan_res * *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.

15.423.2.4 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.

15.423.2.5 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 deauthenticate from the current AP.

15.423.2.6 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.

15.423.2.7 void wpa_supplicant_disable_network (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Mark a configured network as disabled.

Parameters:

wpa_s wpa_supplicant structure for a network interface

ssid wpa_ssid structure for a configured network or NULL

Disables the specified network or all networks if no network specified.

15.423.2.8 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.

15.423.2.9 int wpa_supplicant_driver_init (struct wpa_supplicant * *wpa_s*)

Initialize driver interface parameters.

Parameters:

wpa_s Pointer to wpa_supplicant data

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.

15.423.2.10 void wpa_supplicant_enable_network (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Mark a configured network as enabled.

Parameters:

wpa_s wpa_supplicant structure for a network interface

ssid wpa_ssid structure for a configured network or NULL

Enables the specified network or all networks if no network specified.

15.423.2.11 struct wpa_supplicant* wpa_supplicant_get_iface (struct wpa_global * *global*, const char * *ifname*) [read]

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

15.423.2.12 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_res.

15.423.2.13 struct wpa_ssid* wpa_supplicant_get_ssid (struct wpa_supplicant * *wpa_s*) [read]

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

15.423.2.14 struct wpa_global* wpa_supplicant_init (struct wpa_params * *params*) [read]

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.

15.423.2.15 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.

15.423.2.16 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.

15.423.2.17 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 interfaces when wpa_supplicant is terminated.

15.423.2.18 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.

15.423.2.19 int wpa_supplicant_run (struct wpa_global * *global*)

Run the wpa_supplicant main event loop.

Parameters:

global Pointer to global data from wpa_supplicant_init()

Returns:

0 after successful event loop run, -1 on failure

This function starts the main event loop and continues running as long as there are any remaining events. In most cases, this function is running as long as the wpa_supplicant process in still in use.

15.423.2.20 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); this is the ctx variable registered with struct wpa_driver_ops::init()

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. Most driver interfaces rely on more generic OS mechanism for receiving frames through l2_packet, but if such a mechanism is not available, the driver wrapper may take care of received EAPOL frames and deliver them to the core supplicant code by calling this function.

15.423.2.21 void wpa_supplicant_select_network (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Attempt association with a network.

Parameters:

wpa_s wpa_supplicant structure for a network interface

ssid wpa_ssid structure for a configured network or NULL for any network

15.423.2.22 int wpa_supplicant_set_ap_scan (struct wpa_supplicant * *wpa_s*, int *ap_scan*)

Set AP scan mode for interface.

Parameters:

wpa_s wpa_supplicant structure for a network interface

ap_scan AP scan mode

Returns:

0 if succeed or -1 if ap_scan has an invalid value

15.423.2.23 int wpa_supplicant_set_debug_params (struct wpa_global * *global*, int *debug_level*, int *debug_timestamp*, int *debug_show_keys*)

Set global debug params.

Parameters:

global wpa_global structure

debug_level debug level

debug_timestamp determines if show timestamp in debug data

debug_show_keys determines if show keys in debug data

Returns:

0 if succeed or -1 if debug_level has wrong value

15.423.2.24 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.

15.423.2.25 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.

15.423.2.26 int wpa_supplicant_set_suites (struct wpa_supplicant * *wpa_s*, struct wpa_scan_res * *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).

15.423.2.27 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

15.423.3 Variable Documentation

15.423.3.1 const char* wpa_supplicant_full_license1

Initial value:

```
"This program is free software; you can redistribute it and/or modify\n"
"it under the terms of the GNU General Public License version 2 as\n"
"published by the Free Software Foundation.\n"
"\n"
"This program is distributed in the hope that it will be useful,\n"
"but WITHOUT ANY WARRANTY; without even the implied warranty of\n"
"MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the\n"
"GNU General Public License for more details.\n"
"\n"
```

15.423.3.2 const char* wpa_supplicant_full_license2

Initial value:

```
"You should have received a copy of the GNU General Public License\n"
"along with this program; if not, write to the Free Software\n"
"Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA\n"
"\n"
"Alternatively, this software may be distributed under the terms of the\n"
"BSD license.\n"
"\n"
"Redistribution and use in source and binary forms, with or without\n"
"modification, are permitted provided that the following conditions are\n"
"met:\n"
"\n"
```

15.423.3.3 const char* wpa_supplicant_full_license3

Initial value:

```
"1. Redistributions of source code must retain the above copyright\n"
"   notice, this list of conditions and the following disclaimer.\n"
"\n"
"2. Redistributions in binary form must reproduce the above copyright\n"
"   notice, this list of conditions and the following disclaimer in the\n"
"   documentation and/or other materials provided with the distribution.\n"
"\n"
```

15.423.3.4 const char* wpa_supplicant_full_license4

Initial value:

```
"3. Neither the name(s) of the above-listed copyright holder(s) nor the\n"
"   names of its contributors may be used to endorse or promote products\n"
"   derived from this software without specific prior written permission.\n"
"\n"
"THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS\n"
"\"AS IS\" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT\n"
"LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR\n"
"A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT\n"
```

15.423.3.5 const char* wpa_supplicant_full_license5

Initial value:

```
"OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,\n"
"SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT\n"
"LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,\n"
"DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY\n"
"THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT\n"
"(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE\n"
"OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.\n"
"\n"
```

15.423.3.6 const char* wpa_supplicant_license

Initial value:

```
"This program is free software. You can distribute it and/or modify it\n"
"under the terms of the GNU General Public License version 2.\n"
"\n"
"Alternatively, this software may be distributed under the terms of the\n"
"BSD license. See README and COPYING for more details.\n"
```

15.423.3.7 const char* wpa_supplicant_version

Initial value:

```
"wpa_supplicant v" VERSION_STR "\n"
"Copyright (c) 2003-2009, Jouni Malinen <j@w1.fi> and contributors"
```

15.424 wpa_supplicant/wpa_supplicant_i.h File Reference

wpa_supplicant - Internal definitions `#include "common/defs.h"`

Data Structures

- struct wpa_interface

  Parameters for wpa_supplicant_add_iface().

- struct wpa_params

  Parameters for wpa_supplicant_init().

- struct wpa_global

  Internal, global data for all wpa_supplicant interfaces.

- struct wpa_client_mlme
- struct wpa_supplicant

  Internal data for wpa_supplicant interface.

Defines

- #define **WILDCARD_SSID_SCAN** ((struct wpa_ssid *) 1)

Functions

- int **wpa_set_wep_keys** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)
- 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)

  Initialize driver interface parameters.

- int wpa_supplicant_set_suites (struct wpa_supplicant *wpa_s, struct wpa_scan_res *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_res *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.

- struct 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_enable_network (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  Mark a configured network as enabled.

- void wpa_supplicant_disable_network (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  Mark a configured network as disabled.

- void wpa_supplicant_select_network (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

  Attempt association with a network.

- int wpa_supplicant_set_ap_scan (struct wpa_supplicant *wpa_s, int ap_scan)

  Set AP scan mode for interface.

- int wpa_supplicant_set_debug_params (struct wpa_global *global, int debug_level, int debug_timestamp, int debug_show_keys)

  Set global debug params.

- void **wpa_show_license** (void)
- struct 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.

- struct wpa_supplicant * wpa_supplicant_get_iface (struct wpa_global *global, const char *ifname)
  Get a new network interface.
- struct 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.
- int **wpa_supplicant_enabled_networks** (struct wpa_config *conf)
- 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_notify_scanning** (struct wpa_supplicant *wpa_s, int scanning)
- int **wpa_supplicant_trigger_scan** (struct wpa_supplicant *wpa_s, struct wpa_driver_scan_params *params)
- void **wpa_supplicant_mark_disassoc** (struct wpa_supplicant *wpa_s)

Variables

- const char * **wpa_supplicant_version**
- const char * **wpa_supplicant_license**
- const char * **wpa_supplicant_full_license1**
- const char * **wpa_supplicant_full_license2**
- const char * **wpa_supplicant_full_license3**
- const char * **wpa_supplicant_full_license4**
- const char * **wpa_supplicant_full_license5**

15.424.1 Detailed Description

wpa_supplicant - Internal definitions

Copyright

Copyright (c) 2003-2007, Jouni Malinen <j@w1.fi>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Alternatively, this software may be distributed under the terms of BSD license.

See README and COPYING for more details.

15.424.2 Function Documentation

15.424.2.1 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.

15.424.2.2 struct wpa_supplicant* wpa_supplicant_add_iface (struct wpa_global * *global*, struct wpa_interface * *iface*) [read]

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.

15.424.2.3 void wpa_supplicant_associate (struct wpa_supplicant * *wpa_s*, struct wpa_scan_res * *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.

15.424.2.4 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.

15.424.2.5 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().

15.424.2.6 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 deauthenticate from the current AP.

15.424.2.7 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.

15.424.2.8 void wpa_supplicant_disable_network (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Mark a configured network as disabled.

Parameters:

wpa_s wpa_supplicant structure for a network interface

ssid wpa_ssid structure for a configured network or NULL

Disables the specified network or all networks if no network specified.

15.424.2.9 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.

15.424.2.10 int wpa_supplicant_driver_init (struct wpa_supplicant * *wpa_s*)

Initialize driver interface parameters.

Parameters:

wpa_s Pointer to wpa_supplicant data

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.

15.424.2.11 void wpa_supplicant_enable_network (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Mark a configured network as enabled.

Parameters:

wpa_s wpa_supplicant structure for a network interface

ssid wpa_ssid structure for a configured network or NULL

Enables the specified network or all networks if no network specified.

15.424.2.12 struct wpa_supplicant* wpa_supplicant_get_iface (struct wpa_global * *global*, const char * *ifname*) [read]

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

15.424.2.13 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_res.

15.424.2.14 struct wpa_ssid* wpa_supplicant_get_ssid (struct wpa_supplicant * *wpa_s*) [read]

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

15.424.2.15 struct wpa_global* wpa_supplicant_init (struct wpa_params * *params*) [read]

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.

15.424.2.16 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.

15.424.2.17 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.

15.424.2.18 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 interfaces when wpa_supplicant is terminated.

15.424.2.19 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.

15.424.2.20 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.

15.424.2.21 int wpa_supplicant_run (struct wpa_global * *global*)

Run the wpa_supplicant main event loop.

Parameters:

global Pointer to global data from wpa_supplicant_init()

Returns:

0 after successful event loop run, -1 on failure

This function starts the main event loop and continues running as long as there are any remaining events. In most cases, this function is running as long as the wpa_supplicant process in still in use.

15.424.2.22 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).

15.424.2.23 void wpa_supplicant_select_network (struct wpa_supplicant * *wpa_s*, struct wpa_ssid * *ssid*)

Attempt association with a network.

Parameters:

wpa_s wpa_supplicant structure for a network interface

ssid wpa_ssid structure for a configured network or NULL for any network

15.424.2.24 int wpa_supplicant_set_ap_scan (struct wpa_supplicant * *wpa_s*, int *ap_scan*)

Set AP scan mode for interface.

Parameters:

wpa_s wpa_supplicant structure for a network interface

ap_scan AP scan mode

Returns:

0 if succeed or -1 if ap_scan has an invalid value

15.424.2.25 int wpa_supplicant_set_debug_params (struct wpa_global * *global*, int *debug_level*, int *debug_timestamp*, int *debug_show_keys*)

Set global debug params.

Parameters:

global wpa_global structure

debug_level debug level

debug_timestamp determines if show timestamp in debug data

debug_show_keys determines if show keys in debug data

Returns:

0 if succeed or -1 if debug_level has wrong value

15.424.2.26 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.

15.424.2.27 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.

15.424.2.28 int wpa_supplicant_set_suites (struct wpa_supplicant * *wpa_s*, struct wpa_scan_res * *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).

15.424.2.29 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

15.425 wpa_supplicant/wpas_glue.c File Reference

WPA Supplicant - Glue code to setup EAPOL and RSN modules. `#include "includes.h"`

```
#include "common.h"
#include "eapol_supp/eapol_supp_sm.h"
#include "wpa.h"
#include "eloop.h"
#include "config.h"
#include "l2_packet/l2_packet.h"
#include "wpa_common.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "pmksa_cache.h"
#include "mlme.h"
#include "sme.h"
#include "ieee802_11_defs.h"
#include "wpa_ctrl.h"
#include "wpas_glue.h"
#include "wps_supplicant.h"
```

Functions

- int **wpa_supplicant_init_eapol** (struct wpa_supplicant *wpa_s)
- int **wpa_supplicant_init_wpa** (struct wpa_supplicant *wpa_s)
- void **wpa_supplicant_rsn_supp_set_config** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

15.425.1 Detailed Description

WPA Supplicant - Glue code to setup EAPOL and RSN modules.

Copyright

15.426 wpa_supplicant/wpas_glue.h File Reference

WPA Supplicant - Glue code to setup EAPOL and RSN modules.

Functions

- int **wpa_supplicant_init_eapol** (struct wpa_supplicant *wpa_s)
- int **wpa_supplicant_init_wpa** (struct wpa_supplicant *wpa_s)
- void **wpa_supplicant_rsn_supp_set_config** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid)

15.426.1 Detailed Description

WPA Supplicant - Glue code to setup EAPOL and RSN modules.

Copyright

15.427 wpa_supplicant/wps_supplicant.c File Reference

wpa_supplicant / WPS integration `#include "includes.h"`

```
#include "common.h"
#include "ieee802_11_defs.h"
#include "ieee802_11_common.h"
#include "wpa_common.h"
#include "config.h"
#include "eap_peer/eap.h"
#include "wpa_supplicant_i.h"
#include "driver_i.h"
#include "eloop.h"
#include "uuid.h"
#include "wpa_ctrl.h"
#include "notify.h"
#include "eap_common/eap_wsc_common.h"
#include "blacklist.h"
#include "wpa.h"
#include "wps_supplicant.h"
#include "dh_group5.h"
```

Defines

- #define **WPS_PIN_SCAN_IGNORE_SEL_REG** 3

Functions

- int **wpas_wps_eapol_cb** (struct wpa_supplicant *wpa_s)
- enum wps_request_type **wpas_wps_get_req_type** (struct wpa_ssid *ssid)
- int **wpas_wps_start_pbc** (struct wpa_supplicant *wpa_s, const u8 *bssid)
- int **wpas_wps_start_pin** (struct wpa_supplicant *wpa_s, const u8 *bssid, const char *pin)
- int **wpas_wps_start_reg** (struct wpa_supplicant *wpa_s, const u8 *bssid, const char *pin, struct wps_new_ap_settings *settings)
- int **wpas_wps_init** (struct wpa_supplicant *wpa_s)
- void **wpas_wps_deinit** (struct wpa_supplicant *wpa_s)
- int **wpas_wps_ssid_bss_match** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid, struct wpa_scan_res *bss)
- int **wpas_wps_ssid_wildcard_ok** (struct wpa_supplicant *wpa_s, struct wpa_ssid *ssid, struct wpa_scan_res *bss)
- int **wpas_wps_scan_pbc_overlap** (struct wpa_supplicant *wpa_s, struct wpa_scan_res *selected, struct wpa_ssid *ssid)
- void **wpas_wps_notify_scan_results** (struct wpa_supplicant *wpa_s)
- int **wpas_wps_searching** (struct wpa_supplicant *wpa_s)

- int **wpas_wps_scan_result_text** (const u8 *ies, size_t ies_len, char *buf, char *end)
- int **wpas_wps_er_start** (struct wpa_supplicant *wpa_s)
- int **wpas_wps_er_stop** (struct wpa_supplicant *wpa_s)

15.427.1 Detailed Description

wpa_supplicant / WPS integration

Copyright

15.428 wpa_supplicant/wps_supplicant.h File Reference

wpa_supplicant / WPS integration

15.428.1 Detailed Description

wpa_supplicant / WPS integration

Copyright

Index