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.

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() .

Control interface commands

Following commands can be used with wpa_ctrl_request() :

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.

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
```

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
```

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
```

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
```

SET <variable> <value>

Set variables:

- EAPOL::heldPeriod
- EAPOL::authPeriod

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

Example command:

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

LOGON

IEEE 802.1X EAPOL state machine logon.

LOGOFF

IEEE 802.1X EAPOL state machine logoff.

REASSOCIATE

Force reassociation.

RECONNECT

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

PREAUTH <BSSID>

Start pre-authentication with the given BSSID.

ATTACH

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

DETACH

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

LEVEL <debug level>

Change debug level.

RECONFIGURE

Force wpa_supplicant to re-read its configuration data.

TERMINATE

Terminate wpa_supplicant process.

BSSID <network id> <BSSID>

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

LIST_NETWORKS

List configured networks.

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

(note: fields are separated with tabs)

DISCONNECT

Disconnect and wait for REASSOCIATE or RECONNECT command before connecting.

SCAN

Request a new BSS scan.

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)

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=000b6a6b6d2070726976617465010180dd180050f20101000050f20401000050f20401000050f2020000
ssid=jkm private
```

SELECT_NETWORK <network id>

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

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.

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.

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.

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.

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)
- ...

GET_NETWORK <network id> <variable>

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

SAVE_CONFIG

Save the current configuration.

P2P_FIND

Start P2P device discovery. Optional parameter can be used to specify the duration for the discovery in seconds (e.g., "P2P_FIND 5"). If the duration is not specified, discovery will be started for indefinite time, i.e., until it is terminated by P2P_STOP_FIND or P2P_CONNECT (to start group formation with a discovered peer).

The default search type is to first run a full scan of all channels and then continue scanning only social channels (1, 6, 11). This behavior can be changed by specifying a different search type: social (e.g., "P2P_FIND 5 type=social") will skip the initial full scan and only search social channels; progressive (e.g., "P2P_FIND type=progressive") starts with a full scan and then searches progressively through all channels one channel at the time with the social channel scans. Progressive device discovery can be used to find new groups (and groups that were not found during the initial scan, e.g., due to the GO being asleep) over time without adding considerable extra delay for every Search state round.

P2P_STOP_FIND

Stop ongoing P2P device discovery or other operation (connect, listen mode).

P2P_CONNECT

Start P2P group formation with a discovered P2P peer. This includes group owner negotiation, group interface setup, provisioning, and establishing data connection.

P2P_CONNECT <peer device="" address>=""> <pbc|pin|PIN#> [label|display|keypad] [persistent] [join|auth] [go_intent=<0..15>]

Start P2P group formation with a discovered P2P peer. This includes optional group owner negotiation, group interface setup, provisioning, and establishing data connection.

The <pbc|pin|PIN#> parameter specifies the WPS provisioning method. "pbc" string starts pushbutton method, "pin" string start PIN method using an automatically generated PIN (which will be returned as the command return code), PIN# means that a pre-selected PIN can be used (e.g., 12345670). [label|display|keypad] is used with PIN method to specify which PIN is used (label=PIN from local label, display=dynamically generated random PIN from local display, keypad=PIN entered from peer device label or display). "persistent" parameter can be used to request a persistent group to be formed.

"join" indicates that this is a command to join an existing group as a client. It skips the GO Negotiation part.

"auth" indicates that the WPS parameters are authorized for the peer device without actually starting GO Negotiation (i.e., the peer is expected to initiate GO Negotiation). This is mainly for testing purposes.

The optional "go_intent" parameter can be used to override the default GO Intent value.

P2P_LISTEN

Start Listen-only state. Optional parameter can be used to specify the duration for the Listen operation in seconds. This command may not be of that much use during normal operations and is mainly designed for testing. It can also be used to keep the device discoverable without having to maintain a group.

P2P_GROUP_REMOVE

Terminate a P2P group. If a new virtual network interface was used for the group, it will also be removed. The network interface name of the group interface is used as a parameter for this command.

P2P_GROUP_ADD

Set up a P2P group owner manually (i.e., without group owner negotiation with a specific peer). This is also known as autonomous GO. Optional persistent=<network id>=""> can be used to specify restart of a persistent group.

P2P_PROV_DISC

Send P2P provision discovery request to the specified peer. The parameters for this command are the P2P device address of the peer and the desired configuration method. For example, "P2P_PROV_DISC 02:01:02:03:04:05 display" would request the peer to display a PIN for us and "P2P_PROV_DISC 02:01:02:03:04:05 keypad" would request the peer to enter a PIN that we display.

P2P_GET_PASSPHRASE

Get the passphrase for a group (only available when acting as a GO).

P2P_SERV_DISC_REQ

Schedule a P2P service discovery request. The parameters for this command are the device address of the peer device (or 00:00:00:00:00:00 for wildcard query that is sent to every discovered P2P peer that supports service discovery) and P2P Service Query TLV(s) as hexdump. For example, "P2P_SERV_DISC_REQ 00:00:00:00:00:00 02000001" schedules a request for listing all supported service discovery protocols and requests this to be sent to all discovered peers. The pending requests are sent during device discovery (see P2P_FIND).

This command returns an identifier for the pending query (e.g., "1f77628") that can be used to cancel the request. Directed requests will be automatically removed when the specified peer has replied to it.

P2P_SERV_DISC_CANCEL_REQ

Cancel a pending P2P service discovery request. This command takes a single parameter: identifier for the pending query (the value returned by P2P_SERV_DISC_REQ), e.g., "P2P_SERV_DISC_CANCEL_REQ 1f77628".

P2P_SERV_DISC_RESP

Reply to a service discovery query. This command takes following parameters: frequency in MHz, destination address, dialog token, response TLV(s). The first three parameters are copied from the request event. For example, "P2P_SERV_DISC_RESP 2437 02:40:61:c2:f3:b7 1 0300000101".

P2P_SERVICE_UPDATE

Indicate that local services have changed. This is used to increment the P2P service indicator value so that peers know when previously cached information may have changed.

P2P_SERV_DISC_EXTERNAL

Configure external processing of P2P service requests: 0 (default) = no external processing of requests (i.e., internal code will reject each request), 1 = external processing of requests (external program is responsible for replying to service discovery requests with P2P_SERV_DISC_RESP).

P2P_REJECT

Reject connection attempt from a peer (specified with a device address). This is a mechanism to reject a pending GO Negotiation with a peer and request to automatically block any further connection or discovery of the peer.

P2P_INVITE

Invite a peer to join a group or to (re)start a persistent group.

P2P_PEER

Fetch information about a discovered peer. This command takes in an argument specifying which peer to select: P2P Device Address of the peer, "FIRST" to indicate the first peer in the list, or "NEXT-<P2P Device Address>" to indicate the entry following the specified peer (to allow for iterating through the list).

P2P_EXT_LISTEN

Enable/disable extended listen timing. Without parameters, this command disables extended listen timing. When enabling the feature, two parameters are used: availibility period and availability interval (both in milliseconds and with range of 1-65535).

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
```

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
```

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

INTERFACES

List configured interfaces.

```
wlan0
eth0
```

Control interface events

wpa_supplicant generates number messages based on events like connection or a completion of a task. These are available to external programs that attach to receive unsolicited messages over the control interface with wpa_ctrl_attach() .

The event messages will be delivered over the attach control interface as text strings that start with the priority level of the message and a fixed prefix text as defined in wpa_ctrl.h. After this, optional additional information may be included depending on the event message. For example, following event message is delivered when new scan results are available:

```
<2>CTRL-EVENT-SCAN-RESULTS
```

Following priority levels are used:

- 0 = MSGDUMP
- 1 = DEBUG
- 2 = INFO
- 3 = WARNING
- 4 = ERROR

By default, any priority level greater than equal to 2 (INFO) are delivered over the attached control interface. LEVEL command can be used to set the level of messages which will be delivered. It should be noted that there are many debug messages that do not include any particulat prefix and are subject to change. They may be used for debug information, but can usually be ignored by external programs.

In most cases, the external program can skip over the priority field in the beginning of the event message and then compare the following text to the event strings from wpa_ctrl.h that the program is interested in processing.

Following subsections describe the most common event notifications generated by wpa_supplicant.

CTRL-REQ-

WPA_CTRL_REQ: Request information from a user. See Interactive requests sections for more details.

CTRL-EVENT-CONNECTED

WPA_EVENT_CONNECTED: Indicate successfully completed authentication and that the data connection is now enabled.

CTRL-EVENT-DISCONNECTED

WPA_EVENT_DISCONNECTED: Disconnected, data connection is not available

CTRL-EVENT-TERMINATING

WPA_EVENT_TERMINATING: wpa_supplicant is exiting

CTRL-EVENT-PASSWORD-CHANGED

WPA_EVENT_PASSWORD_CHANGED: Password change was completed successfully

CTRL-EVENT-EAP-NOTIFICATION

WPA_EVENT_EAP_NOTIFICATION: EAP-Request/Notification received

CTRL-EVENT-EAP-STARTED

WPA_EVENT_EAP_STARTED: EAP authentication started (EAP-Request/Identity received)

CTRL-EVENT-EAP-METHOD

WPA_EVENT_EAP_METHOD: EAP method selected

CTRL-EVENT-EAP-SUCCESS

WPA_EVENT_EAP_SUCCESS: EAP authentication completed successfully

CTRL-EVENT-EAP-FAILURE

WPA_EVENT_EAP_FAILURE: EAP authentication failed (EAP-Failure received)

CTRL-EVENT-SCAN-RESULTS

WPA_EVENT_SCAN_RESULTS: New scan results available

CTRL-EVENT-BSS-ADDED

WPA_EVENT_BSS_ADDED: A new BSS entry was added. The event prefix is followed by the BSS entry id and BSSID.

```
CTRL-EVENT-BSS-ADDED 34 00:11:22:33:44:55
```

CTRL-EVENT-BSS-REMOVED

WPA_EVENT_BSS_REMOVED: A BSS entry was removed. The event prefix is followed by BSS entry id and BSSID.

```
CTRL-EVENT-BSS-REMOVED 34 00:11:22:33:44:55
```

WPS-OVERLAP-DETECTED

WPS_EVENT_OVERLAP: WPS overlap detected in PBC mode

WPS-AP-AVAILABLE-PBC

WPS_EVENT_AP_AVAILABLE_PBC: Available WPS AP with active PBC found in scan results.

WPS-AP-AVAILABLE-PIN

WPS_EVENT_AP_AVAILABLE_PIN: Available WPS AP with recently selected PIN registrar found in scan results.

WPS-AP-AVAILABLE

WPS_EVENT_AP_AVAILABLE: Available WPS AP found in scan results

WPS-CRED-RECEIVED

WPS_EVENT_CRED_RECEIVED: A new credential received

WPS-M2D

WPS_EVENT_M2D: M2D received

ctrl_iface_event_WPS_FAIL

WPS_EVENT_FAIL: WPS registration failed after M2/M2D

WPS-SUCCESS

WPS_EVENT_SUCCESS: WPS registration completed successfully

WPS-TIMEOUT

WPS_EVENT_TIMEOUT: WPS enrollment attempt timed out and was terminated

WPS-ENROLLEE-SEEN

WPS_EVENT_ENROLLEE_SEEN: WPS Enrollee was detected (used in AP mode). The event prefix is followed by MAC addr, UUID-E, pri dev type, config methods, dev passwd id, request type, [dev name].

```
WPS-ENROLLEE-SEEN 02:00:00:00:01:00
572cf82f-c957-5653-9b16-b5cfb298abf1 1-0050F204-1 0x80 4 1
[Wireless Client]
```

WPS-ER-AP-ADD

WPS_EVENT_ER_AP_ADD: WPS ER discovered an AP

```
WPS-ER-AP-ADD 87654321-9abc-def0-1234-56789abc0002 02:11:22:33:44:55
pri_dev_type=6-0050F204-1 wps_state=1 |Very friendly name|Company|
Long description of the model|WAP|http://w1.fi/|http://w1.fi/hostapd/
```

WPS-ER-AP-REMOVE

WPS_EVENT_ER_AP_REMOVE: WPS ER removed an AP entry

```
WPS-ER-AP-REMOVE 87654321-9abc-def0-1234-56789abc0002
```

WPS-ER-ENROLLEE-ADD

WPS_EVENT_ER_ENROLLEE_ADD: WPS ER discovered a new Enrollee

```
WPS-ER-ENROLLEE-ADD 2b7093f1-d6fb-5108-adbb-bea66bb87333
02:66:a0:ee:17:27 M1=1 config_methods=0x14d dev_passwd_id=0
pri_dev_type=1-0050F204-1
|Wireless Client|Company|cmodel|123|12345|
```

WPS-ER-ENROLLEE-REMOVE

WPS_EVENT_ER_ENROLLEE_REMOVE: WPS ER removed an Enrollee entry

```
WPS-ER-ENROLLEE-REMOVE 2b7093f1-d6fb-5108-adbb-bea66bb87333
02:66:a0:ee:17:27
```

WPS-PIN-NEEDED

WPS_EVENT_PIN_NEEDED: PIN is needed to complete provisioning with an Enrollee. This is followed by information about the Enrollee (UUID, MAC address, device name, manufacturer, model name, model number, serial number, primary device type).

```
WPS-PIN-NEEDED 5a02a5fa-9199-5e7c-bc46-e183d3cb32f7 02:2a:c4:18:5b:f3
[Wireless Client|Company|cmodel|123|12345|1-0050F204-1]
```

WPS-NEW-AP-SETTINGS

WPS_EVENT_NEW_AP_SETTINGS: New AP settings were received

WPS-REG-SUCCESS

WPS_EVENT_REG_SUCCESS: WPS provisioning was completed successfully (AP/Registrar)

WPS-AP-SETUP-LOCKED

WPS_EVENT_AP_SETUP_LOCKED: AP changed into setup locked state due to multiple failed configuration attempts using the AP PIN.

AP-STA-CONNECTED

AP_STA_CONNECTED: A station associated with us (AP mode event). The event prefix is followed by the MAC address of the station.

```
AP-STA-CONNECTED 02:2a:c4:18:5b:f3
```

AP-STA-DISCONNECTED

AP_STA_DISCONNECTED: A station disassociated (AP mode event)

```
AP-STA-DISCONNECTED 02:2a:c4:18:5b:f3
```

P2P-DEVICE-FOUND

P2P_EVENT_DEVICE_FOUND: Indication of a discovered P2P device with information about that device.

```
P2P-DEVICE-FOUND 02:b5:64:63:30:63 p2p_dev_addr=02:b5:64:63:30:63
pri_dev_type=1-0050f204-1 name='Wireless Client' config_methods=0x84
dev_capab=0x21 group_capab=0x0
```

P2P-GO-NEG-REQUEST

P2P_EVENT_GO_NEG_REQUEST: A P2P device requested GO negotiation, but we were not ready to start the negotiation.

```
P2P-GO-NEG-REQUEST 02:40:61:c2:f3:b7 dev_passwd_id=4
```

P2P-GO-NEG-SUCCESS

P2P_EVENT_GO_NEG_SUCCESS: Indication of successfully complete group owner negotiation.

P2P-GO-NEG-FAILURE

P2P_EVENT_GO_NEG_FAILURE: Indication of failed group owner negotiation.

P2P-GROUP-FORMATION-SUCCESS

P2P_EVENT_GROUP_FORMATION_SUCCESS: Indication that P2P group formation has been completed successfully.

P2P-GROUP-FORMATION-FAILURE

P2P_EVENT_GROUP_FORMATION_FAILURE: Indication that P2P group formation failed (e.g., due to provisioning failure or timeout).

P2P-GROUP-STARTED

P2P_EVENT_GROUP_STARTED: Indication of a new P2P group having been started. Additional parameters: network interface name for the group, role (GO/client), SSID. The passphrase used in the group is also indicated here if known (on GO) or PSK (on client). If the group is a persistent one, a flag indicating that is included.

```
P2P-GROUP-STARTED wlan0-p2p-0 GO ssid="DIRECT-3F Testing"
passphrase="12345678" go_dev_addr=02:40:61:c2:f3:b7 [PERSISTENT]
```

P2P-GROUP-REMOVED

P2P_EVENT_GROUP_REMOVED: Indication of a P2P group having been removed. Additional parameters: network interface name for the group, role (GO/client).

```
P2P-GROUP-REMOVED wlan0-p2p-0 GO
```

P2P-PROV-DISC-SHOW-PIN

P2P_EVENT_PROV_DISC_SHOW_PIN: Request from the peer for us to display a PIN that will be entered on the peer. The following parameters are included after the event prefix: peer_address PIN. The PIN is a random PIN generated for this connection. P2P_CONNECT command can be used to accept the request with the same PIN configured for the connection.

```
P2P-PROV-DISC-SHOW-PIN 02:40:61:c2:f3:b7 12345670
p2p_dev_addr=02:40:61:c2:f3:b7 pri_dev_type=1-0050F204-1 name='Test'
config_methods=0x188 dev_capab=0x21 group_capab=0x0
```

P2P-PROV-DISC-ENTER-PIN

P2P_EVENT_PROV_DISC_ENTER_PIN: Request from the peer for us to enter a PIN displayed on the peer. The following parameter is included after the event prefix: peer address.

```
P2P-PROV-DISC-ENTER-PIN 02:40:61:c2:f3:b7 p2p_dev_addr=02:40:61:c2:f3:b7
pri_dev_type=1-0050F204-1 name='Test' config_methods=0x188
dev_capab=0x21 group_capab=0x0
```

P2P-PROV-DISC-PBC-REQ

P2P_EVENT_PROV_DISC_PBC_REQ: Request from the peer for us to connect using PBC. The following parameters are included after the event prefix: peer_address. P2P_CONNECT command can be used to accept the request.

```
P2P-PROV-DISC-PBC-REQ 02:40:61:c2:f3:b7 p2p_dev_addr=02:40:61:c2:f3:b7
pri_dev_type=1-0050F204-1 name='Test' config_methods=0x188
dev_capab=0x21 group_capab=0x0
```

P2P-PROV-DISC-PBC-RESP

P2P_EVENT_PROV_DISC_PBC_RESP: The peer accepted our provision discovery request to connect using PBC. The following parameters are included after the event prefix: peer_address. P2P_CONNECT command can be used to start GO Negotiation after this.

```
P2P-PROV-DISC-PBC-RESP 02:40:61:c2:f3:b7
```

P2P-SERV-DISC-REQ

P2P-SERV-DISC-REQ: Indicate reception of a P2P service discovery request. The following parameters are included after the event prefix: frequency in MHz, source address, dialog token, Service Update Indicator, Service Query TLV(s) as hexdump.

```
P2P-SERV-DISC-REQ 2412 02:40:61:c2:f3:b7 0 0 02000001
```

P2P-SERV-DISC-RESP

P2P-SERV-DISC-RESP: Indicate reception of a P2P service discovery response. The following parameters are included after the event prefix: source address, Service Update Indicator, Service Response TLV(s) as hexdump.

```
P2P-SERV-DISC-RESP 02:40:61:c2:f3:b7 0 0300000101
```

P2P-INVITATION-RECEIVED

P2P-INVITATION-RECEIVED: Indicate reception of a P2P Invitation Request. For persistent groups, the parameter after the event prefix indicates which network block includes the persistent group data.

```
P2P-INVITATION-RECEIVED sa=02:40:61:c2:f3:b7 persistent=0
```

P2P-INVITATION-RESULT

P2P-INVITATION-RESULT: Indicate result of a P2P invitation that was requested with P2P_INVITE. The parameter status=

shows the status code returned by the peer (or -1 on local failure or timeout).

```
P2P-INVITATION-RESULT status=1
```