OpenSync 2.2 Southbound API

Date: August 7, 2020
Document ID: EDE-019-300-902

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Module Documentation

3.1 OpenSync Target Library

Modules

- Initialization and Cleanup
- Control of Managers
- Interface API
- Ethernet Clients API
- Interface Mapping API
- Certificate Management
- Miscellaneous Overrides
- Radio API
- VIF API
- Statistics Related APIs
- Survey API
- Neighbor Scanning Related API
- Device Info API
- Device Control API
- MAC Learning API
- Client Freeze API
- Band Steering API

Classes

- struct mcproxyd_params

Macros

- #define **TARGET_BUFF_SZ** 256
- #define **TARGET_SERIAL_SZ** OS_MACSTR_PLAIN_SZ
- #define **TARGET_ID_SZ** OS_MACSTR_PLAIN_SZ

Typedefs

- typedef char **ifname**[64]
- typedef struct mcproxyd_params target_mcproxy_params_t

Enumerations

- enum target_prtcl_t {
  DISABLE_IGMP = 1,
  DISABLE_MLD,
  IGMPv1,
  IGMPv2,
  IGMPv3,
  MLDv1,
  MLDv2 }

Functions

- bool target_set_igmp_mcproxy_params (target_mcproxy_params_t *mcparams)
  Applies config to mcproxy and reloads the corresponding daemon.
- bool target_get_igmp_mcproxy_params (target_mcproxy_params_t *mcparams)
  Get config from the mcproxy.
- bool target_set_mld_mcproxy_params (target_mcproxy_params_t *mcparams)
  Applies config to mcproxy and reloads the corresponding daemon.
- bool target_get_mld_mcproxy_params (target_mcproxy_params_t *mcparams)
  Get config from the mcproxy.
- bool target_set_igmp_mcproxy_sys_params (struct schema_IGMP_Config *iccfg)
  Applies mcproxy system parameters and reloads the corresponding proxy daemon.
- bool target_get_igmp_mcproxy_sys_params (struct schema_IGMP_Config *iccfg)
  Get mcproxy system parameters.
- bool target_set_mld_mcproxy_sys_params (struct schema_MLD_Config *mlcfg)
  Applies mcproxy system parameters and reloads the corresponding proxy daemon.
- bool target_get_mld_mcproxy_sys_params (struct schema_MLD_Config *iccfg)
  Get mcproxy system parameters.

3.1.1 Detailed Description

Target API.

3.1.2 Class Documentation

3.1.2.1 struct mcproxyd_params

Multicast Proxy Params required by target_mcproxy_start()

Public Attributes

- target_prtcl_t **protocol**
- char **upstrm_if** [64]
- int **num_dwnstrifs**
- ifname * **dwnstrm_ifs**

3.1.3 Typedef Documentation

3.1.3.1 target_mcproxy_params_t

```
typedef struct mcproxyd_params target_mcproxy_params_t
```

Multicast Proxy Params required by target_mcproxy_start()

3.1.4 Enumeration Type Documentation

3.1.4.1 target_prtcl_t

```
enum target_prtcl_t
```

Multicast proxy value

3.1.5 Function Documentation

3.1.5.1 target_get_igmp_mcproxy_params()

```
bool target_get_igmp_mcproxy_params (
            target_mcproxy_params_t * mcparams )
```

Get config from the mcproxy.

Parameters

mcparams	contains protocol, upstream, and downstream ifs info.

Returns

true on success

3.1.5.2 target_get_igmp_mcproxy_sys_params()

```
bool target_get_igmp_mcproxy_sys_params (
             struct schema_IGMP_Config * iccfg )
```

Get mcproxy system parameters.

Parameters

iccfg	contains all required IGMP params.

Returns

true on success

3.1.5.3 target_get_mld_mcproxy_params()

```
bool target_get_mld_mcproxy_params (
             target_mcproxy_params_t * mcparams )
```

Get config from the mcproxy.

Parameters

mcparams	contains protocol, upstream, and downstream ifs info.

Returns

true on success

3.1.5.4 target_get_mld_mcproxy_sys_params()

```
bool target_get_mld_mcproxy_sys_params (
             struct schema_MLD_Config * iccfg )
```

Get mcproxy system parameters.

Parameters

iccfg	contains all required IGMP params.

Returns

true on success

3.1.5.5 target_set_igmp_mcproxy_params()

```
bool target_set_igmp_mcproxy_params (
             target_mcproxy_params_t * mcparams )
```

Applies config to mcproxy and reloads the corresponding daemon.

Parameters

mcparams	contains protocol, upstream, and downstream ifs info.

Returns

true on success

3.1.5.6 target_set_igmp_mcproxy_sys_params()

```
bool target_set_igmp_mcproxy_sys_params (
             struct schema_IGMP_Config * iccfg )
```

Applies mcproxy system parameters and reloads the corresponding proxy daemon.

Parameters

iccfg	contains all required IGMP params.

Returns

true on success

3.1.5.7 target_set_mld_mcproxy_params()

```
bool target_set_mld_mcproxy_params (
            target_mcproxy_params_t * mcparams )
```

Applies config to mcproxy and reloads the corresponding daemon.

Parameters

mcparams	contains protocol, upstream, and downstream ifs info.

Returns

true on success

3.1.5.8 target_set_mld_mcproxy_sys_params()

```
bool target_set_mld_mcproxy_sys_params (
            struct schema_MLD_Config * mlcfg )
```

Applies mcproxy system parameters and reloads the corresponding proxy daemon.

Parameters

mlcfg	contains all required IGMP params.

Returns

true on success

3.2 Initialization and Cleanup

Enumerations

- enum **target_init_opt_t** {
  TARGET_INIT_COMMON = 0,
  TARGET_INIT_MGR_DM = 1,
  TARGET_INIT_MGR_CM = 2,
  TARGET_INIT_MGR_WM = 3,
  TARGET_INIT_MGR_SM = 4,
  TARGET_INIT_MGR_NM = 5,
  TARGET_INIT_MGR_BM = 6,
  TARGET_INIT_MGR_FM = 7,
  TARGET_INIT_MGR_LM = 8,
  TARGET_INIT_MGR_LEDM = 9,
  TARGET_INIT_MGR_OM = 10,
  TARGET_INIT_MGR_BLEM = 11,
  TARGET_INIT_MGR_QM = 12,
  TARGET_INIT_MGR_PM = 13,
  TARGET_INIT_MGR_FSM = 14,
  TARGET_INIT_MGR_TM = 15,
  TARGET_INIT_MGR_HELLO_WORLD = 16,
  TARGET_INIT_MGR_FCM = 17,
  TARGET_INIT_MGR_PPM = 18,
  TARGET_INIT_MGR_NFM = 19 }

Functions

- bool target_ready (struct ev_loop *loop)

  Wait for platform readiness.
- bool target_init (target_init_opt_t opt, struct ev_loop *loop)

  Perform platform specific initialization.
- bool target_close (target_init_opt_t opt, struct ev_loop *loop)

  Perform platform specific cleanups on exit.

3.2.1 Detailed Description

Target library initialization and cleanup.

3.2.2 Function Documentation

3.2.2.1 target_close()

```
bool target_close (
             target_init_opt_t opt,
             struct ev_loop * loop )
```

Perform platform specific cleanups on exit.

Once the manager exits its processing loop, the target_close API gets called. Example implementation can call cleanup actions of HAL library that was initialized inside target_init(). Note: this function is called from every manager, so cleanup action must correspond to init action for specific manager.

Parameters

opt	specifies from which manager this API is called
loop	main loop handle

Returns

true on success

3.2.2.2 target_init()

```
bool target_init (
            target_init_opt_t opt,
            struct ev_loop * loop )
```

Perform platform specific initialization.

The purpose of target_init is to allow initialization of vendor specific parameters: IOCTL or netlink sockets, linked lists, etc. This API is called from every manager prior to executing the functional APIs. If given vendor specific initialization is not needed for one of the managers (for instance there is no need to have Wifi HAL initialized for DM) it can be a no-op for that manager. Note: each manager is a separate Linux process.

Parameters

opt	specifies from which manager this API is called
loop	main loop handle

Returns

true on success

3.2.2.3 target_ready()

```
bool target_ready (
            struct ev_loop * loop )
```

Wait for platform readiness.

The purpose of target_ready API is to allow the subsystem to fully initialize and is ready for processing. The API needs to be blocking. Successful return starts managers and spawns monitoring.

Example actions that target may want to perform:

- Check if all default network interfaces are setup.
- Check if date and time are set correctly.
- Cache platform data (model, id, version, etc.) for further use.

If the target readiness is assured before starting OpenSync (for example by systemd dependencies) the implementation of this function may just return true.

Parameters

loop	main loop handle

Returns

true on success

3.3 Control of Managers

Classes

- struct target_managers_config_t

Variables

- target_managers_config_t target_managers_config [ ]
  List of managers to start.
- int **target_managers_num**

3.3.1 Detailed Description

Definitions and API related to control of managers.

3.3.2 Class Documentation

3.3.2.1 struct target_managers_config_t

Public Attributes

- char * **name**
- pid_t **pid**
- bool **started**
- int **ordinal**
- int **always_restart**
- int **restart_delay**
- bool **needs_plan_b**

3.3.3 Variable Documentation

3.3.3.1 target_managers_config

```
target_managers_config_t target_managers_config[]
```

List of managers to start.

This defines the subset of managers that DM can start with adding entries to target_managers_config table.

Example:

```
target_managers_config_t target_managers_config[] =
{
    { .name = TARGET_MANAGER_PATH("wm"), .needs_plan_b = true,  },
    { .name = TARGET_MANAGER_PATH("nm"), .needs_plan_b = true,  },
    { .name = TARGET_MANAGER_PATH("cm"), .needs_plan_b = true,  },
    { .name = TARGET_MANAGER_PATH("lm"), .needs_plan_b = true,  },
    { .name = TARGET_MANAGER_PATH("sm"), .needs_plan_b = false, },
}
```

The needs_plan_b parameter is part of the monitoring recovery mechanism where DM restarts ALL managers (true) through target_managers_restart or just particular managers (false).

3.4 Interface API

Functions

- bool target_is_radio_interface_ready (char *phy_name)

  Check if radio interface is ready.
- bool target_is_interface_ready (char *if_name)

  Check if interface is ready.
- const char * target_wan_interface_name ()

  Get wan interface name.

3.4.1 Detailed Description

Definitions and API related to control of interfaces.

3.4.2 Function Documentation

3.4.2.1 target_is_interface_ready()

```
bool target_is_interface_ready (
            char * if_name )
```

Check if interface is ready.

Parameters

if_name	interface name

Returns

true on success

3.4.2.2 target_is_radio_interface_ready()

```
bool target_is_radio_interface_ready (
            char * phy_name )
```

Check if radio interface is ready.

Parameters

phy_name	radio interface name

Returns

true on success

3.4.2.3 target_wan_interface_name()

```
const char* target_wan_interface_name ( )
```

Get wan interface name.

Returns

wan interface name

3.5 Ethernet Clients API

Functions

- const char ** **target_ethclient_iflist_get** ()
- const char ** **target_ethclient_brlist_get** ()

3.5.1 Detailed Description

Definitions and API related to control of ethernet clients.

3.6 Interface Mapping API

Functions

- bool **target_map_init** (void)
- bool **target_map_close** (void)
- bool **target_map_insert** (char ∗if_name, char ∗map_name)
- char ∗ **target_map_ifname** (char ∗ifname)
- bool **target_map_ifname_exists** (const char ∗ifname)
- char ∗ **target_unmap_ifname** (char ∗ifname)
- bool **target_unmap_ifname_exists** (const char ∗ifname)

3.6.1 Detailed Description

API for mapping of interface names that the cloud uses to actual interface names.

This API is deprecated. The actual interface names are defined in a device profile, configured on the cloud.

3.7 Certificate Management

Functions

- const char * target_tls_cacert_filename (void)
  Get the TLS CA certificate filename.
- const char * target_tls_mycert_filename (void)
  Get the TLS certificate filename.
- const char * target_tls_privkey_filename (void)
  Get the TLS private key filename.

3.7.1 Detailed Description

Definitions and API related to control of certificates.

3.7.2 Function Documentation

3.7.2.1 target_tls_cacert_filename()

```
const char* target_tls_cacert_filename (
            void  )
```

Get the TLS CA certificate filename.

Returns

CA filename

3.7.2.2 target_tls_mycert_filename()

```
const char* target_tls_mycert_filename (
            void  )
```

Get the TLS certificate filename.

Returns

certificate filename

3.7.2.3 target_tls_privkey_filename()

```
const char* target_tls_privkey_filename (
            void  )
```

Get the TLS private key filename.

Returns

private key filename

3.8 Miscellaneous Overrides

Functions

- bool target_log_open (char *name, int flags)
  Enable logging.
- bool target_log_pull (const char *upload_location, const char *upload_token)
  Collect logs.
- bool target_log_pull_ext (const char *upload_location, const char *upload_token, const char *upload_method)
  Collect logs (using specified method).
- INTERNAL const char * target_scripts_dir (void)
  Get the directory for scripts.
- const char * target_tools_dir (void)
  Get the directory for tools.
- const char * target_bin_dir (void)
  Get the directory for binaries.
- const char * target_persistent_storage_dir (void)
  Get a persistent storage mount point.
- INTERNAL void target_managers_restart (void)
  Restart all managers.

3.8.1 Detailed Description

API used for various overrides.

3.8.2 Function Documentation

3.8.2.1 target_bin_dir()

```
const char* target_bin_dir (
            void  )
```

Get the directory for binaries.

Returns

binaries directory

3.8.2.2 target_log_open()

```
bool target_log_open (
            char * name,
            int flags )
```

Enable logging.

By default calls log_open. Can be overridden with platform specific functionality.

Parameters

name	name of the binary (manager name)
flags	see flags for log_open()

Returns

true on success

3.8.2.3 target_log_pull()

```
bool target_log_pull (
            const char * upload_location,
            const char * upload_token )
```

Collect logs.

This function will be called upon cloud request. Main job of it is to collect logs (/var/log/messages, dmesg, ...) and system information (interface configuration, running processes, disk usage, ...) and upload them as a single ∗.tgz file to location specified by "upload_location" parameter. Note that ∗.tgz file must be named as a given "upload_token" with .tgz suffix (ie: upload_token.tgz).

Note

Single ∗.tgz file must be smaller than 10MB.
Use of Log Manager is optional

Parameters

upload_location	URL for upload
upload_token	filename to upload

Returns

true on success

3.8.2.4 target_log_pull_ext()

```
bool target_log_pull_ext (
            const char * upload_location,
```

```
            const char * upload_token,
            const char * upload_method )
```

Collect logs (using specified method).

An extended variant of the basic target_log_pull() function.

Parameters

upload_location	URL
upload_token	filename to upload
upload_method	method/procedure required to upload a logpull with the specified type of URL.

Returns

true on success

3.8.2.5 target_managers_restart()

```
INTERNAL void target_managers_restart (
            void  )
```

Restart all managers.

This function restarts all specified managers gracefully. Exact implementation depends on the device and is the responsibility of the vendor. An example might be to call an init.d restart script that performs something similar to stopping and then starting the managers. The script must ensure the system is properly un-initialized and then initialized again, before DM is started again. This includes correct operation and state of ovsdb-server and contents of all tables, removing wireless interfaces, stopping wpa_supplicant, etc.

Returns

true on success

3.8.2.6 target_persistent_storage_dir()

```
const char* target_persistent_storage_dir (
            void  )
```

Get a persistent storage mount point.

Returns

persistent storage directory

3.8.2.7 target_scripts_dir()

```
INTERNAL const char* target_scripts_dir (
            void  )
```

Get the directory for scripts.

Returns

scripts directory

3.8.2.8 target_tools_dir()

```
const char* target_tools_dir (
            void  )
```

Get the directory for tools.

The needed tools inside tools_dir are defined by the cloud controller. For example, the speed test binary is a tool that can be used by the cloud controller. The default implementation can be used by defining CONFIG_TARGET_PATH↩ _TOOLS.

Returns

tools directory

3.9 Radio API

Classes

- struct target_radio_ops

  List of callbacks for radio/vif changes. More...

Functions

- bool target_radio_init (const struct target_radio_ops *ops)

  Hands over WM callbacks so target can notify about vif/radio statuses.
- bool target_radio_config_init2 (void)

  Initialize radio interface configuration.
- bool target_radio_config_need_reset (void)

  Target tells if it requires full re-sync with Config/State.
- bool target_radio_config_set2 (const struct schema_Wifi_Radio_Config *rconf, const struct schema_Wifi_↩ Radio_Config_flags *changed)

  Apply the configuration for the radio interface.
- bool target_radio_state_get (char *ifname, struct schema_Wifi_Radio_State *rstate)

  Get state of radio interface.

3.9.1 Detailed Description

Definitions and API related to control of radios.

3.9.2 Class Documentation

3.9.2.1 struct target_radio_ops

List of callbacks for radio/vif changes.

Public Attributes

- void(* op_vconf)(const struct schema_Wifi_VIF_Config *vconf, const char *phy)
- void(* op_rconf)(const struct schema_Wifi_Radio_Config *rconf)
- void(* op_vstate)(const struct schema_Wifi_VIF_State *vstate, const char *phy)
- void(* op_rstate)(const struct schema_Wifi_Radio_State *rstate)
- void(* op_client)(const struct schema_Wifi_Associated_Clients *client, const char *vif, bool associated)
- void(* op_clients)(const struct schema_Wifi_Associated_Clients *clients, int num, const char *vif)
- void(* op_flush_clients)(const char *vif)

3.9.2.1.1 Member Data Documentation

3.9.2.1.1.1 op_client

```
void(* target_radio_ops::op_client) (const struct schema_Wifi_Associated_Clients *client, const
char *vif, bool associated)
```

target calls this whenever a client connects or disconnects

3.9.2.1.1.2 op_clients

```
void(* target_radio_ops::op_clients) (const struct schema_Wifi_Associated_Clients *clients, int
num, const char *vif)
```

target calls this whenever it wants to re-sync all clients due to, e.g. internal event buffer overrun.

3.9.2.1.1.3 op_flush_clients

```
void(* target_radio_ops::op_flush_clients) (const char *vif)
```

target calls this whenever it wants to clear out all clients on a given vif; intended to use when target wants to fully re-sync connects clients (i.e. the call will be followed by op_client() calls) or when a vif is deconfigured abruptly

3.9.2.1.1.4 op_rconf

```
void(* target_radio_ops::op_rconf) (const struct schema_Wifi_Radio_Config *rconf)
```

target calls this whenever middleware (if exists) wants to update radio configuration

3.9.2.1.1.5 op_rstate

```
void(* target_radio_ops::op_rstate) (const struct schema_Wifi_Radio_State *rstate)
```

target calls this whenever system radio state has changed, e.g. channel changed, target_radio_config_set2() was called

3.9.2.1.1.6 op_vconf

```
void(* target_radio_ops::op_vconf) (const struct schema_Wifi_VIF_Config *vconf, const char *phy)
```

target calls this whenever middleware (if exists) wants to update vif configuration

3.9.2.1.1.7 op_vstate

```
void(* target_radio_ops::op_vstate) (const struct schema_Wifi_VIF_State *vstate, const char *phy)
```

target calls this whenever system vif state has changed, e.g. channel changed, target_vif_config_set2() was called

3.9.3 Function Documentation

3.9.3.1 target_radio_config_init2()

```
bool target_radio_config_init2 (
            void  )
```

Initialize radio interface configuration.

This is called during WM initialization only if target_radio_config_need_reset() is true.

This is expected to call op_rconf and op_vconf with initial radio/vif configuration parameters.

This is intended to handle residential gateways / systems with middleware HAL that can take control over ovsdb.

Returns

true on success.

3.9.3.2 target_radio_config_need_reset()

```
bool target_radio_config_need_reset (
            void  )
```

Target tells if it requires full re-sync with Config/State.

If target implementation talks with a middleware HAL that can sometimes take control over Plume cloud then this function should return true whenever middleware is supposed to be in charge of the wireless configuration.

When true target is expected to call op_vconf and op_rconf during target_radio_config_init2().

Returns

true if middleware exists and target wants target_radio_config_init2() to be called.

3.9.3.3 target_radio_config_set2()

```
bool target_radio_config_set2 (
            const struct schema_Wifi_Radio_Config * rconf,
            const struct schema_Wifi_Radio_Config_flags * changed )
```

Apply the configuration for the radio interface.

This is API v2. Will be called only if target_radio_init() returned true during init.

Parameters

rconf	complete desired radio config
changed	list of fields from rconf that are out of sync with regard to rstate

Returns

true on success, false means the call will be retried later

3.9.3.4 target_radio_init()

```
bool target_radio_init (
             const struct target_radio_ops * ops )
```

Hands over WM callbacks so target can notify about vif/radio statuses.

Target implementation is expected to notify WM about things like channel changes, configuration being applied, clients connecting and disconnecting, etc. via provided callbacks.

Target implementation is free to perform early bookkeeping initialization, e.g. open up sockets to middleware HAL API it talks to, etc.

Returns

true if target is okay. False if it could not initialize. False results in WM using old target API currently. In the future WM will refuse to start if False is returned.

3.9.3.5 target_radio_state_get()

```
bool target_radio_state_get (
             char * ifname,
             struct schema_Wifi_Radio_State * rstate )
```

Get state of radio interface.

This function is used to retrieve the current state of a radio interface

Note

Depending on the implementation, some of the returned values in rstate may be a copy of last applied configuration and not a reflection of the actual interface state

Parameters

ifname	interface name
rstate	output; radio interface state

Returns

true on success

3.10 VIF API

Functions

- bool target_vif_config_set2 (const struct schema_Wifi_VIF_Config *vconf, const struct schema_Wifi_Radio_↩ Config *rconf, const struct schema_Wifi_Credential_Config *cconfs, const struct schema_Wifi_VIF_Config_flags *changed, int num_cconfs)

  Apply the configuration for the vif interface.
- bool target_vif_state_get (char *ifname, struct schema_Wifi_VIF_State *vstate)

  Get state of vif interface.

3.10.1 Detailed Description

Definitions and API related to control of VIFs.

3.10.2 Function Documentation

3.10.2.1 target_vif_config_set2()

```
bool target_vif_config_set2 (
            const struct schema_Wifi_VIF_Config * vconf,
            const struct schema_Wifi_Radio_Config * rconf,
            const struct schema_Wifi_Credential_Config * cconfs,
            const struct schema_Wifi_VIF_Config_flags * changed,
            int num_cconfs )
```

Apply the configuration for the vif interface.

Parameters

vconf	complete desired vif config
rconf	complete desired radio config
cconfs	complete desired vif credential config, used for extender mode to provide multiple network for sta vif
changed	list of fields from vconf that are out of sync with state
num_cconfs	number of cconfs entries

Returns

true on success, false means the call will be retried later

3.10.2.2 target_vif_state_get()

```
bool target_vif_state_get (
             char * ifname,
             struct schema_Wifi_VIF_State * vstate )
```

Get state of vif interface.

This function is used to retrieve the current state of a vif interface

Note

Depending on the implementation, some of the returned values in vstate may be a copy of last applied configuration and not a reflection of the actual interface state

Parameters

ifname	interface name
vstate	output; vif interface state

Returns

true on success

3.11 Statistics Related APIs

Typedefs

- typedef bool **target_stats_clients_cb_t**(ds_dlist_t *client_list, void *ctx, int status)

Functions

- bool target_radio_tx_stats_enable (radio_entry_t *radio_cfg, bool status)

  Enable radio tx stats.
- bool target_radio_fast_scan_enable (radio_entry_t *radio_cfg, ifname_t if_name)

  Enable radio fast scan.
- target_client_record_t * **target_client_record_alloc** ()
- void **target_client_record_free** (target_client_record_t *record)
- bool target_stats_clients_get (radio_entry_t *radio_cfg, radio_essid_t *essid, target_stats_clients_cb_t *client_cb, ds_dlist_t *client_list, void *client_ctx)

  Get clients stats.
- bool target_stats_clients_convert (radio_entry_t *radio_cfg, target_client_record_t *client_list_new, target_client_record_t *client_list_old, dpp_client_record_t *client_record)

  Calculate client stats deltas.

3.11.1 Detailed Description

Definitions and API related to statistics.

3.11.2 Function Documentation

3.11.2.1 target_radio_fast_scan_enable()

```
bool target_radio_fast_scan_enable (
            radio_entry_t * radio_cfg,
            ifname_t if_name )
```

Enable radio fast scan.

Parameters

radio_cfg	radio interface handle
if_name	radio interface name

Returns

true on success

3.11.2.2 target_radio_tx_stats_enable()

```
bool target_radio_tx_stats_enable (
             radio_entry_t * radio_cfg,
             bool status )
```

Enable radio tx stats.

Parameters

radio_cfg	radio interface handle
status	true (enable) or false (disable)

Returns

true on success

3.11.2.3 target_stats_clients_convert()

```
bool target_stats_clients_convert (
             radio_entry_t * radio_cfg,
             target_client_record_t * client_list_new,
             target_client_record_t * client_list_old,
             dpp_client_record_t * client_record )
```

Calculate client stats deltas.

Calculates the deltas between new and old client list and stores the result into client_record

Parameters

radio_cfg	radio interface handle
client_list_new	new values
client_list_old	old values
client_record	output; calculated deltas

Returns

true on success

3.11.2.4 target_stats_clients_get()

```
bool target_stats_clients_get (
             radio_entry_t * radio_cfg,
             radio_essid_t * essid,
             target_stats_clients_cb_t * client_cb,
             ds_dlist_t * client_list,
             void * client_ctx )
```

Get clients stats.

The results will be provided to the callback function and can be called either synchronously or asynchronously depending on platform specifics

Parameters

radio_cfg	radio interface handle
essid	SSID string
client_cb	callback function
client_list	output; resulting client list
client_ctx	optional context for callback

Returns

true on success

3.12 Survey API

Typedefs

- typedef bool **target_stats_survey_cb_t**(ds_dlist_t *survey_list, void *survey_ctx, int status)

Functions

- target_survey_record_t * **target_survey_record_alloc** ()
- void **target_survey_record_free** (target_survey_record_t *record)
- bool target_stats_survey_get (radio_entry_t *radio_cfg, uint32_t *chan_list, uint32_t chan_num, radio_scan_type_t scan_type, target_stats_survey_cb_t *survey_cb, ds_dlist_t *survey_list, void *survey_ctx)

  Get radio channel survey stats.
- bool target_stats_survey_convert (radio_entry_t *radio_cfg, radio_scan_type_t scan_type, target_survey_record_t *data_new, target_survey_record_t *data_old, dpp_survey_record_t *survey_record)

  Calculate channel survey deltas.

3.12.1 Detailed Description

Definitions and API related to surveys.

3.12.2 Function Documentation

3.12.2.1 target_stats_survey_convert()

```
bool target_stats_survey_convert (
            radio_entry_t * radio_cfg,
            radio_scan_type_t scan_type,
            target_survey_record_t * data_new,
            target_survey_record_t * data_old,
            dpp_survey_record_t * survey_record )
```

Calculate channel survey deltas.

Calculates the deltas between new and old channel survey and stores the result into survey_record

Parameters

radio_cfg	radio interface handle
scan_type	scan type
data_new	new values
data_old	old values
survey_record	output; calculated deltas

Returns

true on success

3.12.2.2 target_stats_survey_get()

```
bool target_stats_survey_get (
             radio_entry_t * radio_cfg,
             uint32_t * chan_list,
             uint32_t chan_num,
             radio_scan_type_t scan_type,
             target_stats_survey_cb_t * survey_cb,
             ds_dlist_t * survey_list,
             void * survey_ctx )
```

Get radio channel survey stats.

The results will be provided to the callback function and can be called either synchronously or asynchronously depending on platform specifics

Parameters

radio_cfg	radio interface handle
chan_list	list of channels
chan_num	number of channels in list
scan_type	scan type
survey_cb	callback function
survey_list	output; survey stats
survey_ctx	optional context for callback

Returns

true on success

3.13 Neighbor Scanning Related API

Typedefs

- typedef bool **target_scan_cb_t**(void ∗scan_ctx, int status)

Functions

- bool target_stats_scan_start (radio_entry_t ∗radio_cfg, uint32_t ∗chan_list, uint32_t chan_num, radio_scan_↩ type_t scan_type, int32_t dwell_time, target_scan_cb_t ∗scan_cb, void ∗scan_ctx)

  Start neighbor scan.
- bool target_stats_scan_stop (radio_entry_t ∗radio_cfg, radio_scan_type_t scan_type)

  Stop neighbor scan.
- bool target_stats_scan_get (radio_entry_t ∗radio_cfg, uint32_t ∗chan_list, uint32_t chan_num, radio_scan_↩ type_t scan_type, dpp_neighbor_report_data_t ∗scan_results)

  Get neighbor stats.

3.13.1 Detailed Description

Definitions and API related to neighbor scanning.

3.13.2 Function Documentation

3.13.2.1 target_stats_scan_get()

```
bool target_stats_scan_get (
            radio_entry_t * radio_cfg,
            uint32_t * chan_list,
            uint32_t chan_num,
            radio_scan_type_t scan_type,
            dpp_neighbor_report_data_t * scan_results )
```

Get neighbor stats.

Parameters

radio_cfg	radio interface handle
chan_list	channel list
chan_num	number of channels
scan_type	scan type
scan_results	output; neighbor stats

Returns

true on success

3.13.2.2 target_stats_scan_start()

```
bool target_stats_scan_start (
            radio_entry_t * radio_cfg,
            uint32_t * chan_list,
            uint32_t chan_num,
            radio_scan_type_t scan_type,
            int32_t dwell_time,
            target_scan_cb_t * scan_cb,
            void * scan_ctx )
```

Start neighbor scan.

The scanning will be performed in background and the callback function will be called when the results are available. The actual results need to be fetched with target_stats_scan_get()

Parameters

radio_cfg	radio interface handle
chan_list	channel list
chan_num	number of channels
scan_type	scan type
dwell_time	dwell time in ms
scan_cb	callback function
scan_ctx	optional context for callback

Returns

true on success

3.13.2.3 target_stats_scan_stop()

```
bool target_stats_scan_stop (
            radio_entry_t * radio_cfg,
            radio_scan_type_t scan_type )
```

Stop neighbor scan.

Parameters

radio_cfg	radio interface handle
scan_type	scan type

Returns

true on success

3.14 Device Info API

Functions

- bool target_stats_device_get (dpp_device_record_t *device_entry)

  Get device stats.
- bool target_stats_device_temp_get (radio_entry_t *radio_cfg, dpp_device_temp_t *device_entry)

  Get device temperature.
- bool target_stats_device_txchainmask_get (radio_entry_t *radio_cfg, dpp_device_txchainmask_t *txchainmask_entry)

  Get device txchainmask.
- bool target_stats_device_fanrpm_get (uint32_t *fan_rpm)

  Get device fan RPM.

3.14.1 Detailed Description

Definitions and API related to device information.

3.14.2 Function Documentation

3.14.2.1 target_stats_device_fanrpm_get()

```
bool target_stats_device_fanrpm_get (
            uint32_t * fan_rpm )
```

Get device fan RPM.

Parameters

fan_rpm	RPM of the internal fan

Returns

true on success

3.14.2.2 target_stats_device_get()

```
bool target_stats_device_get (
            dpp_device_record_t * device_entry )
```

Get device stats.

Returns device load average (loadavg) and uptime

Parameters

device_entry	output; device stats

Returns

true on success

3.14.2.3 target_stats_device_temp_get()

```
bool target_stats_device_temp_get (
            radio_entry_t * radio_cfg,
            dpp_device_temp_t * device_entry )
```

Get device temperature.

Parameters

radio_cfg	radio interface handle
device_entry	output; device stats

Returns

true on success

3.14.2.4 target_stats_device_txchainmask_get()

```
bool target_stats_device_txchainmask_get (
            radio_entry_t * radio_cfg,
            dpp_device_txchainmask_t * txchainmask_entry )
```

Get device txchainmask.

Parameters

radio_cfg	radio interface handle
txchainmask_entry	txchainmask of device

Returns

true on success

3.15 Device Control API

Classes

- struct target_connectivity_check_t

Macros

- #define TARGET_GW_TYPE (1 << 0)
- #define TARGET_EXTENDER_TYPE (1 << 1)

Enumerations

- enum target_connectivity_check_option_t {
  LINK_CHECK = 1 << 0,
  ROUTER_CHECK = 1 << 1,
  INTERNET_CHECK = 1 << 2,
  NTP_CHECK = 1 << 3 }

Functions

- bool target_device_config_register (void *awlan_cb)

  Subscribe to changes of device config.
- bool target_device_config_set (struct schema_AWLAN_Node *awlan)

  Apply device config.
- bool target_device_execute (const char *cmd)

  Execute external tools.
- int target_device_capabilities_get ()

  Get device capabilities.
- bool target_device_connectivity_check (const char *ifname, target_connectivity_check_t *cstate, target_connectivity_check_option_t opts)

  Get device connectivity status.
- bool target_device_restart_managers ()

  Restart plume managers.
- bool target_device_wdt_ping ()

  Ping watchdog system.

3.15.1 Detailed Description

Definitions and API related to device control.

3.15.2 Class Documentation

3.15.2.1 struct target_connectivity_check_t

States returned by target_device_connectivity_check()

Public Attributes

- bool link_state

  If link has an IP, the link_state should be set to 'true' if it can be pinged. Otherwise a custom (vendor-specific) way of checking link state must be provided.
- bool router_state

  True if the IP of default gateway can be pinged.
- bool internet_state

  True if external IP address can be pinged.
- bool ntp_state

  True if current datetime is set correctly.

3.15.3 Macro Definition Documentation

3.15.3.1 TARGET_EXTENDER_TYPE

```
#define TARGET_EXTENDER_TYPE (1 << 1)
```

returned by target_device_capabilities_get()

3.15.3.2 TARGET_GW_TYPE

```
#define TARGET_GW_TYPE (1 << 0)
```

returned by target_device_capabilities_get()

3.15.4 Enumeration Type Documentation

3.15.4.1 target_connectivity_check_option_t

```
enum target_connectivity_check_option_t
```

Option flags for target_device_connectivity_check()

3.15.5 Function Documentation

3.15.5.1 target_device_capabilities_get()

```
int target_device_capabilities_get ( )
```

Get device capabilities.

All targets are at least TARGET_GW_TYPE, so example implementation can return just TARGET_GW_TYPE. If the target is also capable of being an extender, the TARGET_EXTENDER_TYPE should be set in a bitmask additionally.

Returns

device capabilities as a bitmask based on target capabilities types

3.15.5.2 target_device_config_register()

```
bool target_device_config_register (
            void * awlan_cb )
```

Subscribe to changes of device config.

This is for changes of device config that originate from external management protocols not ovsdb. The changes will then be applied to ovsdb by the callback. The device config is a data described inside AWLAN_Node table. The example implementation may want to set custom cloud redirector address here and call the awlan_cb() whenever the redirector address is updated. If the redirector address is static and the target is not going to update any other field of AWLAN_Node table it is safe to make this function a no-op.

callback type: void (∗update)(struct schema_AWLAN_Node ∗awlan, schema_filter_t ∗filter);

Parameters

awlan_cb	callback function

Returns

true on success

3.15.5.3 target_device_config_set()

```
bool target_device_config_set (
             struct schema_AWLAN_Node * awlan )
```

Apply device config.

This applies device config from ovsdb to external management protocols (if available). The device config is a data described inside AWLAN_Node table. Example field of that table that may need to be synchronized with target-specific implementation is a 'device_mode'. If target doesn't need to perform any action when the content of this table is updated then it is safe to make this function a no-op.

Parameters

awlan	ovsdb schema for AWLAN_node table.

Returns

true on success

3.15.5.4 target_device_connectivity_check()

```
bool target_device_connectivity_check (
             const char * ifname,
             target_connectivity_check_t * cstate,
             target_connectivity_check_option_t opts )
```

Get device connectivity status.

For example implementation, see target_kconfig.c

Parameters

ifname	interface name
cstate	connectivity state
opts	which checks to perform

Returns

true if all links are in correct state, false otherwise.

3.15.5.5 target_device_execute()

```
bool target_device_execute (
            const char * cmd )
```

Execute external tools.

The implementation of this function should provide ability to run a shell command.

Parameters

cmd	command string

Returns

true on success

3.15.5.6 target_device_restart_managers()

```
bool target_device_restart_managers ( )
```

Restart plume managers.

Returns

true on success

3.15.5.7 target_device_wdt_ping()

```
bool target_device_wdt_ping ( )
```

Ping watchdog system.

If the target provides a watchdog that checks if OpenSync managers are alive, the implementation of this function should feed that watchdog. If target doesn't use such functionality it's safe to just return true.

Returns

true on success

3.16 MAC Learning API

Typedefs

- typedef bool target_mac_learning_cb_t(struct schema_OVS_MAC_Learning *omac, bool oper_status)
  Ethernet client change callback type.

Functions

- bool target_mac_learning_register (target_mac_learning_cb_t *omac_cb)
  Subscribe to ethernet client change events.

3.16.1 Detailed Description

Definitions and API related to MAC learning.

3.16.2 Function Documentation

3.16.2.1 target_mac_learning_register()

```
bool target_mac_learning_register (
            target_mac_learning_cb_t * omac_cb )
```

Subscribe to ethernet client change events.

Parameters

omac_cb	a callback function

Returns

true on success

3.17 Client Freeze API

Typedefs

- typedef bool **target_client_nickname_cb_t**(struct schema_Client_Nickname_Config ∗cncfg, bool status)
- typedef bool **target_client_freeze_cb_t**(struct schema_Client_Freeze_Config ∗cfcfg, bool status)

Functions

- bool **target_client_nickname_register** (target_client_nickname_cb_t ∗nick_cb)
- bool **target_client_nickname_set** (struct schema_Client_Nickname_Config ∗cncfg)
- bool **target_client_freeze_register** (target_client_freeze_cb_t ∗freze_cb)
- bool **target_client_freeze_set** (struct schema_Client_Freeze_Config ∗cfcfg)

3.17.1 Detailed Description

Definitions and API related to Client Freeze functionality.

3.18 Band Steering API

Classes

- struct bsal_ifconfig_t
- struct bsal_client_config_t
- struct bsal_neigh_info_t
- struct bsal_btm_params_t
- struct bsal_rrm_params_t
- struct bsal_datarate_info_t
- struct bsal_rrm_caps_t
- struct bsal_ev_probe_req_t
- struct bsal_ev_connect_t
- struct bsal_ev_disconnect_t
- struct bsal_ev_activity_t
- struct bsal_ev_chan_util_t
- struct bsal_ev_rssi_xing_t
- struct bsal_ev_rssi_t
- struct bsal_ev_steer_t
- struct bsal_ev_auth_fail_t
- struct bsal_ev_action_frame_t
- struct bsal_event_t
- struct bsal_client_info_t

Macros

- #define **BSAL_IFNAME_LEN** 17
- #define **BSAL_MAC_ADDR_LEN** 6
- #define **BSAL_MAX_TM_NEIGHBORS** 3
- #define **BSAL_MAX_ASSOC_IES_LEN** 1024
- #define **BSAL_MAX_ACTION_FRAME_LEN** 1024

Typedefs

- typedef void(∗ **bsal_event_cb_t**) (bsal_event_t ∗event)

Enumerations

- enum bsal_ev_type_t {
  BSAL_EVENT_PROBE_REQ = 1,
  BSAL_EVENT_CLIENT_CONNECT,
  BSAL_EVENT_CLIENT_DISCONNECT,
  BSAL_EVENT_CLIENT_ACTIVITY,
  BSAL_EVENT_CHAN_UTILIZATION,
  BSAL_EVENT_RSSI_XING,
  BSAL_EVENT_RSSI,
  BSAL_EVENT_STEER_CLIENT,
  BSAL_EVENT_STEER_SUCCESS,
  BSAL_EVENT_STEER_FAILURE,
  BSAL_EVENT_AUTH_FAIL,
  BSAL_EVENT_ACTION_FRAME,
  BSAL_EVENT_DEBUG_CHAN_UTIL = 128,
  BSAL_EVENT_DEBUG_RSSI }

- enum **bsal_disc_source_t** {
  BSAL_DISC_SOURCE_LOCAL = 0,
  BSAL_DISC_SOURCE_REMOTE }
- enum **bsal_disc_type_t** {
  BSAL_DISC_TYPE_DISASSOC = 0,
  BSAL_DISC_TYPE_DEAUTH }
- enum **bsal_rssi_change_t** {
  BSAL_RSSI_UNCHANGED = 0,
  BSAL_RSSI_HIGHER,
  BSAL_RSSI_LOWER }
- enum **bsal_phy_mode_t** {
  BSAL_PHY_MODE_AUTO = 0,
  BSAL_PHY_MODE_11A = 1,
  BSAL_PHY_MODE_11B = 2,
  BSAL_PHY_MODE_11G = 3,
  BSAL_PHY_MODE_FH = 4,
  BSAL_PHY_MODE_TURBO_A = 5,
  BSAL_PHY_MODE_TURBO_G = 6,
  BSAL_PHY_MODE_11NA_HT20 = 7,
  BSAL_PHY_MODE_11NG_HT20 = 8,
  BSAL_PHY_MODE_11NA_HT40PLUS = 9,
  BSAL_PHY_MODE_11NA_HT40MINUS = 10,
  BSAL_PHY_MODE_11NG_HT40PLUS = 11,
  BSAL_PHY_MODE_11NG_HT40MINUS = 12,
  BSAL_PHY_MODE_11NG_HT40 = 13,
  BSAL_PHY_MODE_11NA_HT40 = 14,
  BSAL_PHY_MODE_11AC_VHT20 = 15,
  BSAL_PHY_MODE_11AC_VHT40PLUS = 16,
  BSAL_PHY_MODE_11AC_VHT40MINUS = 17,
  BSAL_PHY_MODE_11AC_VHT40 = 18,
  BSAL_PHY_MODE_11AC_VHT80 = 19,
  BSAL_PHY_MODE_11AC_VHT160 = 20,
  BSAL_PHY_MODE_11AC_VHT80_80 = 21 }
- enum **bsal_max_chwidth_t** {
  BSAL_MAX_CHWIDTH_20MHZ = 0,
  BSAL_MAX_CHWIDTH_40MHZ = 1,
  BSAL_MAX_CHWIDTH_80MHZ = 2,
  BSAL_MAX_CHWIDTH_160MHZ = 3 }

Functions

- int target_bsal_init (bsal_event_cb_t event_cb, struct ev_loop ∗loop)

  Gives target a chance to hook and initialize ∗ internals.
- int target_bsal_cleanup (void)

  Gives target a chance to clean up on shutdown.
- int target_bsal_iface_add (const bsal_ifconfig_t ∗ifcfg)

  Requests target to start managing provided interface.
- int target_bsal_iface_update (const bsal_ifconfig_t ∗ifcfg)

  Requests target to update configuration on already managed interface.
- int target_bsal_iface_remove (const bsal_ifconfig_t ∗ifcfg)

  Requests target to stop managing provided interface.
- int target_bsal_client_add (const char ∗ifname, const uint8_t ∗mac_addr, const bsal_client_config_t ∗conf)

Requests target to start managing provided client.

- int target_bsal_client_update (const char *ifname, const uint8_t *mac_addr, const bsal_client_config_t *conf)

  Requests target to update provided client policy configuration.
- int target_bsal_client_remove (const char *ifname, const uint8_t *mac_addr)

  Requests target to stop managing a client.
- int target_bsal_client_measure (const char *ifname, const uint8_t *mac_addr, int num_samples)

  Requests target to schedule signal strength measurement.
- int target_bsal_client_disconnect (const char *ifname, const uint8_t *mac_addr, bsal_disc_type_t type, uint8_t reason)

  Requests target to disconnect a client.
- int target_bsal_client_info (const char *ifname, const uint8_t *mac_addr, bsal_client_info_t *info)

  Requests target to provide client capabilities.
- int target_bsal_bss_tm_request (const char *ifname, const uint8_t *mac_addr, const bsal_btm_params_t *btm_params)

  Requests target to send a BSS Transition Request frame.
- int target_bsal_rrm_beacon_report_request (const char *ifname, const uint8_t *mac_addr, const bsal_rrm_params_t *rrm_params)

  Requests target to send RRM Beacon Report Request frame.
- int target_bsal_rrm_set_neighbor (const char *ifname, const bsal_neigh_info_t *nr)

  Requests target to add an entry to neighbor list.
- int target_bsal_rrm_remove_neighbor (const char *ifname, const bsal_neigh_info_t *nr)

  Requests target to remove an entry from neighbor list.
- int target_bsal_send_action (const char *ifname, const uint8_t *mac_addr, const uint8_t *data, unsigned int data_len)

  Request target to send action frame.

3.18.1 Detailed Description

3.18.2 Class Documentation

3.18.2.1 struct bsal_ifconfig_t

Public Attributes

- char **ifname** [BSAL_IFNAME_LEN]
- uint8_t **chan_util_check_sec**
- uint8_t **chan_util_avg_count**
- uint8_t **inact_check_sec**
- uint8_t **inact_tmout_sec_normal**
- uint8_t **inact_tmout_sec_overload**
- uint8_t **def_rssi_inact_xing**
- uint8_t **def_rssi_low_xing**
- uint8_t **def_rssi_xing**
- struct {
  bool **raw_chan_util**
  bool **raw_rssi**
  } **debug**

3.18.2.2 struct bsal_client_config_t

Public Attributes

- bool **blacklist**
- uint8_t **rssi_probe_hwm**
- uint8_t **rssi_probe_lwm**
- uint8_t **rssi_auth_hwm**
- uint8_t **rssi_auth_lwm**
- uint8_t **rssi_inact_xing**
- uint8_t **rssi_high_xing**
- uint8_t **rssi_low_xing**
- uint8_t **auth_reject_reason**

3.18.2.3 struct bsal_neigh_info_t

Public Attributes

- uint8_t **bssid** [BSAL_MAC_ADDR_LEN]
- uint32_t **bssid_info**
- uint8_t **op_class**
- uint8_t **channel**
- uint8_t **phy_type**
- uint8_t **opt_subelems** [64]
- uint8_t **opt_subelems_len**

3.18.2.4 struct bsal_btm_params_t

Public Attributes

- bsal_neigh_info_t **neigh** [BSAL_MAX_TM_NEIGHBORS]
- int **num_neigh**
- uint8_t **valid_int**
- uint8_t **abridged**
- uint8_t **pref**
- uint8_t **disassoc_imminent**
- uint16_t **bss_term**
- int **tries**
- int **max_tries**
- int **retry_interval**
- bool **inc_neigh**
- bool **inc_self**

3.18.2.5 struct bsal_rrm_params_t

Public Attributes

- uint8_t **op_class**
- uint8_t **channel**
- uint8_t **rand_ivl**
- uint8_t **meas_dur**
- uint8_t **meas_mode**
- uint8_t **req_ssid**
- uint8_t **rep_cond**
- uint8_t **rpt_detail**
- uint8_t **req_ie**
- uint8_t **chanrpt_mode**

3.18.2.6 struct bsal_datarate_info_t

Public Attributes

- bsal_max_chwidth_t **max_chwidth**
- uint8_t **max_streams**
- bsal_phy_mode_t **phy_mode**
- uint8_t **max_MCS**
- uint8_t **max_txpower**
- uint8_t **is_static_smps**
- uint8_t **is_mu_mimo_supported**

3.18.2.7 struct bsal_rrm_caps_t

Public Attributes

- bool **link_meas**
- bool **neigh_rpt**
- bool **bcn_rpt_passive**
- bool **bcn_rpt_active**
- bool **bcn_rpt_table**
- bool **lci_meas**
- bool **ftm_range_rpt**

3.18.2.8 struct bsal_ev_probe_req_t

Public Attributes

- uint8_t **client_addr** [BSAL_MAC_ADDR_LEN]
- uint8_t **rssi**
- bool **ssid_null**
- bool **blocked**

3.18.2.9 struct bsal_ev_connect_t

Public Attributes

- uint8_t **client_addr** [BSAL_MAC_ADDR_LEN]
- uint8_t **assoc_ies** [1024]
- uint8_t **is_BTM_supported**
- uint8_t **is_RRM_supported**
- bool **band_cap_2G**
- bool **band_cap_5G**
- bsal_datarate_info_t **datarate_info**
- bsal_rrm_caps_t **rrm_caps**
- size_t **assoc_ies_len**

3.18.2.10 struct bsal_ev_disconnect_t

Public Attributes

- uint8_t **client_addr** [BSAL_MAC_ADDR_LEN]
- uint8_t **reason**
- bsal_disc_source_t **source**
- bsal_disc_type_t **type**

3.18.2.11 struct bsal_ev_activity_t

Public Attributes

- uint8_t **client_addr** [BSAL_MAC_ADDR_LEN]
- bool **active**

3.18.2.12 struct bsal_ev_chan_util_t

Public Attributes

- uint8_t **utilization**

3.18.2.13 struct bsal_ev_rssi_xing_t

Public Attributes

- uint8_t **client_addr** [BSAL_MAC_ADDR_LEN]
- uint8_t **rssi**
- bsal_rssi_change_t **inact_xing**
- bsal_rssi_change_t **high_xing**
- bsal_rssi_change_t **low_xing**

3.18.2.14 struct bsal_ev_rssi_t

Public Attributes

- uint8_t **client_addr** [BSAL_MAC_ADDR_LEN]
- uint8_t **rssi**

3.18.2.15 struct bsal_ev_steer_t

Public Attributes

- uint8_t **client_addr** [BSAL_MAC_ADDR_LEN]
- int **from_ch**
- int **to_ch**
- uint8_t **rssi**
- bool **connected**

3.18.2.16 struct bsal_ev_auth_fail_t

Public Attributes

- uint8_t **client_addr** [BSAL_MAC_ADDR_LEN]
- uint8_t **rssi**
- uint8_t **reason**
- uint8_t **bs_blocked**
- uint8_t **bs_rejected**

3.18.2.17 struct bsal_ev_action_frame_t

Public Attributes

- uint8_t **data** [BSAL_MAX_ACTION_FRAME_LEN]
- unsigned int **data_len**

3.18.2.18 struct bsal_event_t

Public Attributes

- bsal_ev_type_t **type**
- char **ifname** [BSAL_IFNAME_LEN]
- uint64_t **timestamp_ms**
- union {
  bsal_ev_probe_req_t **probe_req**
  bsal_ev_connect_t **connect**
  bsal_ev_disconnect_t **disconnect**
  bsal_ev_activity_t **activity**
  bsal_ev_chan_util_t **chan_util**
  bsal_ev_rssi_xing_t **rssi_change**
  bsal_ev_rssi_t **rssi**
  bsal_ev_steer_t **steer**
  bsal_ev_auth_fail_t **auth_fail**
  bsal_ev_action_frame_t **action_frame**
  } **data**

3.18.2.19 struct bsal_client_info_t

Public Attributes

- bool **connected**
- uint8_t **is_BTM_supported**
- uint8_t **is_RRM_supported**
- bool **band_cap_2G**
- bool **band_cap_5G**
- bsal_datarate_info_t **datarate_info**
- bsal_rrm_caps_t **rrm_caps**
- uint8_t **assoc_ies** [BSAL_MAX_ASSOC_IES_LEN]
- uint16_t **assoc_ies_len**
- uint8_t **snr**
- uint64_t **tx_bytes**
- uint64_t **rx_bytes**

3.18.3 Enumeration Type Documentation

3.18.3.1 bsal_ev_type_t

```
enum bsal_ev_type_t
```

Enumerator

BSAL_EVENT_CLIENT_ACTIVITY	station started, or stopped traffic
BSAL_EVENT_CHAN_UTILIZATION	deprecated
BSAL_EVENT_RSSI_XING	station rssi crossed lwm or hwm, if configured
BSAL_EVENT_RSSI	see target_bsal_client_measure()
BSAL_EVENT_STEER_CLIENT	deprecated
BSAL_EVENT_STEER_SUCCESS	deprecated
BSAL_EVENT_STEER_FAILURE	deprecated
BSAL_EVENT_AUTH_FAIL	reported when station is rejected by driver
BSAL_EVENT_ACTION_FRAME	received action frame
BSAL_EVENT_DEBUG_CHAN_UTIL	deprecated
BSAL_EVENT_DEBUG_RSSI	deprecated

3.18.4 Function Documentation

3.18.4.1 target_bsal_bss_tm_request()

```
int target_bsal_bss_tm_request (
            const char * ifname,
            const uint8_t * mac_addr,
            const bsal_btm_params_t * btm_params )
```

Requests target to send a BSS Transition Request frame.

Note

target_bsal_client_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
mac_addr	6-byte MAC address of the client
btm_params	BSS Transition Request parameters to use

Returns

0 is treated as success, anything else is an error

3.18.4.2 target_bsal_cleanup()

```
int target_bsal_cleanup (
            void  )
```

Gives target a chance to clean up on shutdown.

Returns

0 is treated as success, anything else is an error

3.18.4.3 target_bsal_client_add()

```
int target_bsal_client_add (
            const char * ifname,
            const uint8_t * mac_addr,
            const bsal_client_config_t * conf )
```

Requests target to start managing provided client.

Note

target shall start calling event_cb() for this client
target_bsal_iface_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
mac_addr	6-byte MAC address of the client
conf	Client policy details

Returns

0 is treated as success, anything else is an error

3.18.4.4 target_bsal_client_disconnect()

```
int target_bsal_client_disconnect (
            const char * ifname,
            const uint8_t * mac_addr,
            bsal_disc_type_t type,
            uint8_t reason )
```

Requests target to disconnect a client.

Note

target_bsal_client_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
mac_addr	6-byte MAC address of the client
type	Whether to send Deauth or Disassoc 802.11 frame
reason	What reason code to use for Deauth or Disassoc 802.11 frame

Returns

0 is treated as success, anything else is an error

3.18.4.5 target_bsal_client_info()

```
int target_bsal_client_info (
            const char * ifname,
            const uint8_t * mac_addr,
            bsal_client_info_t * info )
```

Requests target to provide client capabilities.

Parameters

ifname	Wireless interface name the client is connected on
mac_addr	6-byte MAC address of the client
info	Output buffer where the capabilities shall be put into

Returns

0 is treated as success, anything else is an error

3.18.4.6 target_bsal_client_measure()

```
int target_bsal_client_measure (
            const char * ifname,
            const uint8_t * mac_addr,
            int num_samples )
```

Requests target to schedule signal strength measurement.

The target is expected to generate BSAL_EVENT_RSSI asynchronously after this function returns.

Note

target_bsal_client_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
mac_addr	6-byte MAC address of the client
num_samples	Number of samples to average from. Single RSSI samples tend to vary a lot so it's often desired to collect more than one and smooth it. The target is left with the decision about the algorithm.

Returns

0 is treated as success, anything else is an error

3.18.4.7 target_bsal_client_remove()

```
int target_bsal_client_remove (
            const char * ifname,
            const uint8_t * mac_addr )
```

Requests target to stop managing a client.

Note

target_bsal_client_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
mac_addr	6-byte MAC address of the client

Returns

0 is treated as success, anything else is an error

3.18.4.8 target_bsal_client_update()

```
int target_bsal_client_update (
            const char * ifname,
            const uint8_t * mac_addr,
            const bsal_client_config_t * conf )
```

Requests target to update provided client policy configuration.

Note

target_bsal_client_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
mac_addr	6-byte MAC address of the client
conf	Client policy details

Returns

0 is treated as success, anything else is an error

3.18.4.9 target_bsal_iface_add()

```
int target_bsal_iface_add (
            const bsal_ifconfig_t * ifcfg )
```

Requests target to start managing provided interface.

This may involve setting up a unix socket, or a netlink socket, preparing a bpf filter, calling helper libs or interacting with the driver.

The target is not expected to be calling event_cb() from target_bsal_init() yet. It is safe for it do it, but it will be ignored. It makes sense after first target_bsal_client_add() is called.

Parameters

ifcfg	Interface configuration details

Returns

0 is treated as success, anything else is an error

3.18.4.10 target_bsal_iface_remove()

```
int target_bsal_iface_remove (
            const bsal_ifconfig_t * ifcfg )
```

Requests target to stop managing provided interface.

Note

target_bsal_iface_add() will be called sometime earlier first

Parameters

ifcfg	Interface configuration details

Returns

0 is treated as success, anything else is an error

3.18.4.11 target_bsal_iface_update()

```
int target_bsal_iface_update (
            const bsal_ifconfig_t * ifcfg )
```

Requests target to update configuration on already managed interface.

This may involve setting up a unix socket, or a netlink socket, preparing a bpf filter, calling helper libs or interacting with the driver.

Note

target_bsal_iface_add() will be called sometime earlier first

Parameters

ifcfg	Interface configuration details

Returns

0 is treated as success, anything else is an error

3.18.4.12 target_bsal_init()

```
int target_bsal_init (
            bsal_event_cb_t event_cb,
            struct ev_loop * loop )
```

Gives target a chance to hook and initialize * internals.

Parameters

event_cb	Target shall call this to report events back to BM. It is safe to call this from other threads. Thread safety is a subject to change in the future so using threads is highly discouraged. Target implementation is free to use ev_async over provided mainloop.
loop	Mainloop pointer of BM. Target can use it as long as it accesses it from the same thread the function was originally called from.

Returns

0 is treated as success, anything else is an error

3.18.4.13 target_bsal_rrm_beacon_report_request()

```
int target_bsal_rrm_beacon_report_request (
            const char * ifname,
            const uint8_t * mac_addr,
            const bsal_rrm_params_t * rrm_params )
```

Requests target to send RRM Beacon Report Request frame.

This is used to force a client to generate Probe Request traffic so that both other radios on the AP, as well as other mesh APs, can check the signal strength of other possible links that could be set up.

Note

target_bsal_client_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
mac_addr	6-byte MAC address of the client
rrm_params	RRM Beacon Report Request

Returns

0 is treated as success, anything else is an error

3.18.4.14 target_bsal_rrm_remove_neighbor()

```
int target_bsal_rrm_remove_neighbor (
            const char * ifname,
            const bsal_neigh_info_t * nr )
```

Requests target to remove an entry from neighbor list.

This is required for AP to handle BTM Query Request frames, ie. when clients actively seek roaming information from the AP they are connected to.

Note

target_bsal_iface_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
nr	Neighbor info

Returns

0 is treated as success, anything else is an error

3.18.4.15 target_bsal_rrm_set_neighbor()

```
int target_bsal_rrm_set_neighbor (
            const char * ifname,
            const bsal_neigh_info_t * nr )
```

Requests target to add an entry to neighbor list.

This is required for AP to handle BTM Query Request frames, ie. when clients actively seek roaming information from the AP they are connected to.

Note

target_bsal_iface_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
nr	Neighbor info

Returns

0 is treated as success, anything else is an error

3.18.4.16 target_bsal_send_action()

```
int target_bsal_send_action (
            const char * ifname,
            const uint8_t * mac_addr,
            const uint8_t * data,
            unsigned int data_len )
```

Request target to send action frame.

This is new way to handle BTM, RRM, NR.

Note

target_bsal_client_add() will be called sometime earlier first

Parameters

ifname	Wireless interface name the client is connected on
mac_addr	6-byte MAC address of the client
data	action frame buffer
data_len	action frame buffer length

3.19 OpenSync Networking

Modules

- Common API and Types
- IPv4
- IPv6
- L2 Interface
- PPPoE
- VLAN

3.19.1 Detailed Description

OpenSync Networking APIs

3.20 Common API and Types

Modules

- osn_ip_addr_t
- osn_ip6_addr_t
- osn_mac_addr_t

3.20.1 Detailed Description

Common OpenSync Networking API and types

3.21 osn_ip_addr_t

Classes

- struct osn_ip_addr

Macros

- #define OSN_IP_ADDR_INIT
- #define OSN_IP_ADDR_LEN sizeof("255.255.255.255/32")
- #define PRI_osn_ip_addr "%s"
- #define FMT_osn_ip_addr(x) (__FMT_osn_ip_addr((char[OSN_IP_ADDR_LEN]){0}, OSN_IP_ADDR_LEN, &x))

Typedefs

- typedef struct osn_ip_addr osn_ip_addr_t

Functions

- char * __FMT_osn_ip_addr (char *buf, size_t sz, const osn_ip_addr_t *addr)
- bool osn_ip_addr_from_str (osn_ip_addr_t *out, const char *str)
- bool osn_ip_addr_from_in_addr (osn_ip_addr_t *out, const struct in_addr *in)
- bool osn_ip_addr_from_sockaddr (osn_ip_addr_t *out, const struct sockaddr *in)
- int osn_ip_addr_cmp (void *a, void *b)
- osn_ip_addr_t osn_ip_addr_subnet (osn_ip_addr_t *addr)
- osn_ip_addr_t osn_ip_addr_to_bcast (osn_ip_addr_t *addr)
- int osn_ip_addr_to_prefix (osn_ip_addr_t *addr)
- osn_ip_addr_t osn_ip_addr_from_prefix (int prefix)

3.21.1 Detailed Description

IPv4 Address types and associated functions.

3.21.2 Class Documentation

3.21.2.1 struct osn_ip_addr

IPv4 address definition; this includes the netmask (prefix).

A negative prefix indicates that the prefix is not present. Note that a prefix of 0 is valid (for example, the default route).

This structure should not be accessed directly.

Use OSN_IP_ADDR_INIT to initialize this structure.

Public Attributes

- struct in_addr ia_addr
- int ia_prefix

3.21.2.1.1 Member Data Documentation

3.21.2.1.1.1 ia_addr

```
struct in_addr osn_ip_addr::ia_addr
```

IPv4 Address

3.21.2.1.1.2 ia_prefix

```
int osn_ip_addr::ia_prefix
```

Netmask in /XX notation

3.21.3 Macro Definition Documentation

3.21.3.1 FMT_osn_ip_addr

```
#define FMT_osn_ip_addr(
              x ) (__FMT_osn_ip_addr((char[OSN_IP_ADDR_LEN]){0}, OSN_IP_ADDR_LEN, &x))
```

Macro helper for printf() formatting. See PRI_osn_ip_addr for more info.

3.21.3.2 OSN_IP_ADDR_INIT

```
#define OSN_IP_ADDR_INIT
```

Value:

```
(osn_ip_addr_t)    \
{                                  \
    .ia_prefix = -1,               \
}
```

Initializer for an IPv4 address structure (osn_ip_addr_t)

3.21.3.3 OSN_IP_ADDR_LEN

```
#define OSN_IP_ADDR_LEN sizeof("255.255.255.255/32")
```

Maximum length of a IPv4 Address structure when expressed as a string, including the terminating \0

3.21.3.4 PRI_osn_ip_addr

```
#define PRI_osn_ip_addr "%s"
```

Macro helpers for printf() formatting. The PRI_ macro can be used in conjunction with the FMT_ macro to print IPv4 addresses.

Examples:

```
osn_ip_addr_t my_ipaddr;

printf("Hello. The IP address is: "PRI_osn_ip_addr"\n",
      FMT_osn_ip_addr(my_ipaddr));
```

3.21.4 Typedef Documentation

3.21.4.1 osn_ip_addr_t

```
typedef struct osn_ip_addr osn_ip_addr_t
```

IPv4 address definition; this includes the netmask (prefix).

A negative prefix indicates that the prefix is not present. Note that a prefix of 0 is valid (for example, the default route).

This structure should not be accessed directly.

Use OSN_IP_ADDR_INIT to initialize this structure.

3.21.5 Function Documentation

3.21.5.1 __FMT_osn_ip_addr()

```
char* __FMT_osn_ip_addr (
            char * buf,
            size_t sz,
            const osn_ip_addr_t * addr )
```

Macro helper for printf() formatting. See PRI_osn_ip_addr for more info.

3.21.5.2 osn_ip_addr_cmp()

```
int osn_ip_addr_cmp (
            void * a,
            void * b )
```

Comparator for osn_ip_addr_t structures.

Parameters

in	*a*	First osn_ip_addr_t to compare
in	*b*	Second osn_ip_addr_t to compare

Returns

This function returns an integer less than, equal to, or greater than zero if a is found, respectively, to be less than, to match, or be greater than b.

3.21.5.3 osn_ip_addr_from_in_addr()

```
bool osn_ip_addr_from_in_addr (
             osn_ip_addr_t * out,
             const struct in_addr * in )
```

Initialize a osn_ip_addr_t structure from a in_addr structure. in_addr is commonly used hidden inside struct sockaddr

3.21.5.4 osn_ip_addr_from_prefix()

```
osn_ip_addr_t osn_ip_addr_from_prefix (
             int prefix )
```

Convert a prefix integer to an IP representation. For example:

```
24 -> 255.255.255.0
```

Parameters

in	*prefix*	Prefix to convert

Returns

This function returns an osn_ip_addr_t structure representing the prefix

3.21.5.5 osn_ip_addr_from_sockaddr()

```
bool osn_ip_addr_from_sockaddr (
             osn_ip_addr_t * out,
             const struct sockaddr * in )
```

Initialize a son_ip_addr_t structure from a sockaddr structure.

3.21.5.6 osn_ip_addr_from_str()

```
bool osn_ip_addr_from_str (
            osn_ip_addr_t * out,
            const char * str )
```

Initialize an osn_ip_addr_t from a string. Valid string formats are:

"NN.NN.NN.NN"

or

"NN.NN.NN.NN/NN"

Parameters

in	*out*	Output osn_ip_addr_t structure
in	*str*	Input string

Returns

This function returns true if `str` is valid and was successfully parsed, false otherwise. If false is returned, `out` should be considered invalid.

3.21.5.7 osn_ip_addr_subnet()

```
osn_ip_addr_t osn_ip_addr_subnet (
            osn_ip_addr_t * addr )
```

Strip the non-subnet part of an IP address. For example:

```
192.168.40.1/24 -> 192.168.40.0/24
```

Parameters

in	*addr*	Address to convert

Returns

Returns an osn_ip_addr_t structure that has its non-subnet part set to all zeroes

3.21.5.8 osn_ip_addr_to_bcast()

```
osn_ip_addr_t osn_ip_addr_to_bcast (
            osn_ip_addr_t * addr )
```

Calculate a broadcast address from the given address in `addr`

```
192.168.40.1/24 -> 192.168.40.255
```

Parameters

`in`	*addr*	Address to convert

Returns

This function returns a valid broadcast address with the prefix part removed

3.21.5.9 osn_ip_addr_to_prefix()

```
int osn_ip_addr_to_prefix (
            osn_ip_addr_t * addr )
```

Converts a subnet IP representation to a prefix integer. For example:

```
255.255.255.0 -> 24
```

Parameters

`in`	*addr*	Input address

Returns

Returns the number of consecutive bits set in `addr`

3.22 osn_ip6_addr_t

Classes

- struct osn_ip6_addr

Macros

- #define OSN_IP6_ADDR_INIT
- #define OSN_IP6_ADDR_LEN sizeof("1111:2222:3333:4444:5555:6666:7777:8888/128,2147483648,2147483648")
- #define PRI_osn_ip6_addr "%s"
- #define FMT_osn_ip6_addr(x) (__FMT_osn_ip6_addr((char[OSN_IP6_ADDR_LEN]){0}, OSN_IP6_ADDR_LEN, &x))

Typedefs

- typedef struct osn_ip6_addr osn_ip6_addr_t

Functions

- char * __FMT_osn_ip6_addr (char *buf, size_t sz, const osn_ip6_addr_t *addr)
- bool osn_ip6_addr_from_str (osn_ip6_addr_t *out, const char *str)
- int osn_ip6_addr_cmp (void *a, void *b)
- int osn_ip6_addr_nolft_cmp (void *_a, void *_b)

3.22.1 Detailed Description

IPv6 Address types and associated functions.

3.22.2 Class Documentation

3.22.2.1 struct osn_ip6_addr

IPv6 Address definition; this includes the prefix and lifetimes.

If the prefix is -1, it should be considered not present.

If a lifetime is set to INT_MIN, it should be considered absent, while a value of -1 means infinite.

Use OSN_IP6_ADDR_INIT to initialize this structure to default values.

Public Attributes

- struct in6_addr ia6_addr
- int ia6_prefix
- int ia6_pref_lft
- int ia6_valid_lft

3.22.2.1.1 Member Data Documentation

3.22.2.1.1.1 ia6_addr

```
struct in6_addr osn_ip6_addr::ia6_addr
```

Global IP address

3.22.2.1.1.2 ia6_pref_lft

```
int osn_ip6_addr::ia6_pref_lft
```

Preferred lifetime in seconds (INT_MIN means not set)

3.22.2.1.1.3 ia6_prefix

```
int osn_ip6_addr::ia6_prefix
```

IP prefix – usually 64

3.22.2.1.1.4 ia6_valid_lft

```
int osn_ip6_addr::ia6_valid_lft
```

Valid lifetime in seconds (INT_MIN means not set)

3.22.3 Macro Definition Documentation

3.22.3.1 FMT_osn_ip6_addr

```
#define FMT_osn_ip6_addr(
              x ) (__FMT_osn_ip6_addr((char[OSN_IP6_ADDR_LEN]){0}, OSN_IP6_ADDR_LEN, &x))
```

Macro helper for printf() formatting. See PRI_osn_ip6_addr for more info.

3.22.3.2 OSN_IP6_ADDR_INIT

```
#define OSN_IP6_ADDR_INIT
```

Value:

```
(osn_ip6_addr_t)  \
{                                       \
    .ia6_prefix = -1,                   \
    .ia6_pref_lft = INT_MIN,            \
    .ia6_valid_lft = INT_MIN,           \
}
```

Initializer for an IPv6 address structure (osn_ip6_addr_t)

3.22.3.3 OSN_IP6_ADDR_LEN

```
#define OSN_IP6_ADDR_LEN sizeof("1111:2222:3333:4444:5555:6666:7777:8888/128,2147483648,2147483648")
```

Maximum length of IPv6 Address structure when expressed as a string, including the terminating \0

3.22.3.4 PRI_osn_ip6_addr

```
#define PRI_osn_ip6_addr "%s"
```

Macro helpers for printf() formatting. The PRI_ macro can be used in conjunction with the FMT_ macro to print IPv6 addresses.

Examples:

```
osn_ip6_addr_t my_ip6addr;

printf("Hello. The IPv6 address is: "PRI_osn_ip6_addr"\n",
      FMT_osn_ip6_addr(my_ipaddr));
```

3.22.4 Typedef Documentation

3.22.4.1 osn_ip6_addr_t

```
typedef struct osn_ip6_addr osn_ip6_addr_t
```

IPv6 Address definition; this includes the prefix and lifetimes.

If the prefix is -1, it should be considered not present.

If a lifetime is set to INT_MIN, it should be considered absent, while a value of -1 means infinite.

Use OSN_IP6_ADDR_INIT to initialize this structure to default values.

3.22.5 Function Documentation

3.22.5.1 __FMT_osn_ip6_addr()

```
char* __FMT_osn_ip6_addr (
            char * buf,
            size_t sz,
            const osn_ip6_addr_t * addr )
```

Macro helper for printf() formatting. See PRI_osn_ip6_addr for more info.

3.22.5.2 osn_ip6_addr_cmp()

```
int osn_ip6_addr_cmp (
            void * a,
            void * b )
```

Comparator for osn_ip6_addr_t structures.

Parameters

in	*a*	First osn_ip6_addr_t to compare
in	*b*	Second osn_ip6_addr_t to compare

Returns

This function returns an integer less than, equal to, or greater than zero if a is found, respectively, to be less than, to match, or be greater than b.

3.22.5.3 osn_ip6_addr_from_str()

```
bool osn_ip6_addr_from_str (
            osn_ip6_addr_t * out,
            const char * str )
```

Initialize an osn_ip6_addr_t from a string. Valid string formats are:

IPV6_ADDR/PREFIX,MIN_LFT,MAX_LFT

- IPV6_ADDR - Anything that inet_pton(AF_INET6, ...) can understand

- PREFIX - An integer between 1 and 64 bits, a value of -1 means that the prefix is not present
- MIN_LFT - An integer representing the minimum lifetime in seconds
- MAX_LFT - An integer representing the maximum lifetime in seconds

A value of -1 for MIN_LFT and MAX_LFT means infinite lifetime. A value of INT_MIN for MIN_LFT and MAX_LFT means that the lifetime is not present.

Parameters

in	*out*	Output osn_ip6_addr_t structure
in	*str*	Input string

Returns

This function returns true if `str` is valid and was successfully parsed, false otherwise. If false is returned, `out` should be considered invalid.

3.22.5.4 osn_ip6_addr_nolft_cmp()

```
int osn_ip6_addr_nolft_cmp (
            void * _a,
            void * _b )
```

Comparator for osn_ip6_addr_t structures. This version ignores the IPv6 address lifetimes.

Parameters

in	_a	First osn_ip6_addr_t to compare
in	_b	Second osn_ip6_addr_t to compare

Returns

This function returns an integer less than, equal to, or greater than zero if `a` is found, respectively, to be less than, to match, or be greater than `b`.

3.23 osn_mac_addr_t

Classes

- struct osn_mac_addr

Macros

- #define OSN_MAC_ADDR_LEN sizeof("11:22:33:44:55:66")
- #define OSN_MAC_ADDR_INIT (osn_mac_addr_t){ .ma_addr = { 0 }, }
- #define PRI_osn_mac_addr "%02x:%02x:%02x:%02x:%02x:%02x"
- #define FMT_osn_mac_addr(x)

Typedefs

- typedef struct osn_mac_addr osn_mac_addr_t

Functions

- bool osn_mac_addr_from_str (osn_mac_addr_t *out, const char *str)
- int osn_mac_addr_cmp (void *_a, void *_b)

3.23.1 Detailed Description

Hardware Address (MAC) types and associated functions.

3.23.2 Class Documentation

3.23.2.1 struct osn_mac_addr

MAC address definition. It is advisable that this structure is never used directly but through osn_mac_addr_* functions.

Public Attributes

- uint8_t ma_addr [6]

3.23.2.1.1 Member Data Documentation

3.23.2.1.1.1 ma_addr

```
uint8_t osn_mac_addr::ma_addr[6]
```

Raw MAC address bytes

3.23.3 Macro Definition Documentation

3.23.3.1 FMT_osn_mac_addr

```
#define FMT_osn_mac_addr(
               x )
```

Value:

```
(x).ma_addr[0], \
                                (x).ma_addr[1], \
                                (x).ma_addr[2], \
                                (x).ma_addr[3], \
                                (x).ma_addr[4], \
                                (x).ma_addr[5]
```

Macro helper for printf() formatting. See PRI_osn_mac_addr for more info.

3.23.3.2 OSN_MAC_ADDR_INIT

```
#define OSN_MAC_ADDR_INIT (osn_mac_addr_t){ .ma_addr = { 0 }, }
```

Initializer for a MAC address structure (osn_mac_addr_t)

3.23.3.3 OSN_MAC_ADDR_LEN

```
#define OSN_MAC_ADDR_LEN sizeof("11:22:33:44:55:66")
```

Maximum length of MAC Address structure when expressed as a string, including the terminating \0

This structure should be initialized with OSN_MAC_ADDR_INIT before use.

3.23.3.4 PRI_osn_mac_addr

```
#define PRI_osn_mac_addr "%02x:%02x:%02x:%02x:%02x:%02x"
```

Macro helpers for printf() formatting. The PRI_ macro can be used in conjunction with the FMT_ macro to print MAC addresses.

Examples:

```
osn_mac_addr_t my_hwaddr;

printf("Hello. The MAC address is: "PRI_osn_mac_addr"\n",
      FMT_osn_mac_addr(my_macaddr));
```

3.23.4 Typedef Documentation

3.23.4.1 osn_mac_addr_t

```
typedef struct osn_mac_addr osn_mac_addr_t
```

MAC address definition. It is advisable that this structure is never used directly but through osn_mac_addr_∗ functions.

3.23.5 Function Documentation

3.23.5.1 osn_mac_addr_cmp()

```
int osn_mac_addr_cmp (
            void * _a,
            void * _b )
```

Comparator for osn_mac_addr_t structures.

Parameters

in	↩ _↩ *a*	First osn_mac_addr_t to compare
in	↩ _↩ *b*	Second osn_mac_addr_t to compare

Returns

This function returns an integer less than, equal to, or greater than zero if `a` is found, respectively, to be less than, to match, or be greater than `b`.

3.23.5.2 osn_mac_addr_from_str()

```
bool osn_mac_addr_from_str (
            osn_mac_addr_t * out,
            const char * str )
```

Initialize an osn_mac_addr_t from a string. Valid string formats are:

"XX:XX:XX:XX:XX:XX"

Parameters

in	*out*	Output osn_mac_addr_t structure
in	*str*	Input string

Returns

This function returns true if `str` is valid and was successfully parsed, false otherwise. If false is returned, `out` should be considered invalid.

3.24 IPv4

Modules

- IPv4 Routing
- DHCPv4
- UPnP

Classes

- struct osn_ip_status

Typedefs

- typedef struct osn_ip osn_ip_t
- typedef void osn_ip_status_fn_t(osn_ip_t *ip, struct osn_ip_status *status)

Functions

- osn_ip_t * osn_ip_new (const char *ifname)
- bool osn_ip_del (osn_ip_t *ip)
- bool osn_ip_addr_add (osn_ip_t *ip, const osn_ip_addr_t *addr)
- bool osn_ip_addr_del (osn_ip_t *ip, const osn_ip_addr_t *addr)
- bool osn_ip_dns_add (osn_ip_t *ip, const osn_ip_addr_t *dns)
- bool osn_ip_dns_del (osn_ip_t *ip, const osn_ip_addr_t *dns)
- bool osn_ip_route_gw_add (osn_ip_t *ip, const osn_ip_addr_t *src, const osn_ip_addr_t *gw)
- bool osn_ip_route_gw_del (osn_ip_t *ip, const osn_ip_addr_t *src, const osn_ip_addr_t *gw)
- void osn_ip_data_set (osn_ip_t *ip, void *data)
- void * osn_ip_data_get (osn_ip_t *ip)
- void osn_ip_status_notify (osn_ip_t *ip, osn_ip_status_fn_t *fn)
- bool osn_ip_apply (osn_ip_t *ip)

3.24.1 Detailed Description

OpenSync IPv4 API

3.24.2 Class Documentation

3.24.2.1 struct osn_ip_status

IPv4 status structure. A structure of this type is used when reporting the status of the IPv4 object. See osn_ip_status↩ _fn_t() and osn_ip_status_notify() for more details.

Public Attributes

- const char * is_ifname
- size_t is_addr_len
- osn_ip_addr_t * is_addr
- size_t is_dns_len
- osn_ip_addr_t * is_dns

3.24.2.1.1 Member Data Documentation

3.24.2.1.1.1 is_addr

```
osn_ip_addr_t* osn_ip_status::is_addr
```

List of IPv4 addresses on interface

3.24.2.1.1.2 is_addr_len

```
size_t osn_ip_status::is_addr_len
```

Length of is_addr array

3.24.2.1.1.3 is_dns

```
osn_ip_addr_t* osn_ip_status::is_dns
```

List of DNS servers

3.24.2.1.1.4 is_dns_len

```
size_t osn_ip_status::is_dns_len
```

Length of is_dns array

3.24.2.1.1.5 is_ifname

```
const char* osn_ip_status::is_ifname
```

Interface name

3.24.3 Typedef Documentation

3.24.3.1 osn_ip_status_fn_t

```
typedef void osn_ip_status_fn_t(osn_ip_t *ip, struct osn_ip_status *status)
```

osn_ip_t status notification callback type

A function of this type, registered via osn_ip_status_notify, will be invoked whenever the osn_ip_t object wishes to report the IPv4 status.

Typically this will happen whenever an IPv4 status change is detected (for example, when the IP of the interface changes).

Some implementations may choose to call this function periodically even if there has been no status change detected.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object
in	*status*	A pointer to a osn_ip_status

3.24.3.2 osn_ip_t

```
typedef struct osn_ip osn_ip_t
```

OSN IPv4 object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_ip_new() and must be destroyed using osn_ip_del().

3.24.4 Function Documentation

3.24.4.1 osn_ip_addr_add()

```
bool osn_ip_addr_add (
            osn_ip_t * ip,
            const osn_ip_addr_t * addr )
```

Add an IPv4 address to the IPv4 object.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object
in	*addr*	A pointer to a valid IPv4 address (osn_ip_addr_t)

Note

The new configuration may not take effect until osn_ip_apply() is called.
If osn_ip_addr_add() returns success when adding a duplicate address then osn_ip_addr_del() should return success when removing an invalid address.

3.24.4.2 osn_ip_addr_del()

```
bool osn_ip_addr_del (
            osn_ip_t * ip,
            const osn_ip_addr_t * addr )
```

Remove an IPv4 address from the IPv4 object.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object
in	*addr*	A pointer to a valid IPv4 address (osn_ip_addr_t)

Note

The new configuration may not take effect until osn_ip_apply() is called.

3.24.4.3 osn_ip_apply()

```
bool osn_ip_apply (
            osn_ip_t * ip )
```

Ensure that all configuration pertaining the `self` object is applied to the running system.

How the configuration is applied to the system is highly implementation dependent. Sometimes it makes sense to cluster together several configuration parameters (for example, dnsmasq uses a single config file).

osn_ip_apply() makes sure that a write operation is initiated for all currently cached (dirty) configuration data.

Note

It is not guaranteed that the configuration will be applied as soon as osn_ip_apply() returns – only that the configuration process will be started for all pending operations.

3.24.4.4 osn_ip_data_get()

```
void* osn_ip_data_get (
            osn_ip_t * ip )
```

Get the IPv4 user data.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object

Returns

This function returns the data that was previously set using osn_ip_data_set().

3.24.4.5 osn_ip_data_set()

```
void osn_ip_data_set (
            osn_ip_t * ip,
            void * data )
```

Set the IPv4 user data.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object
in	*data*	Private data

3.24.4.6 osn_ip_del()

```
bool osn_ip_del (
            osn_ip_t * ip )
```

Destroy a valid osn_ip_t object.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

Note

All resources that were allocated during the lifetime of the object are freed. The implementation may choose to remove all IPv4 addresses regardless if they were added using osn_ip_addr_add().
If osn_ip_addr_add() returns success when adding a duplicate address then osn_ip_addr_del() should return success when removing an invalid address.

3.24.4.7 osn_ip_dns_add()

```
bool osn_ip_dns_add (
             osn_ip_t * ip,
             const osn_ip_addr_t * dns )
```

Add an DNSv4 server IP to the IPv4 object.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object
in	*dns*	A pointer to a valid DNSv4 address (osn_ip_addr_t)

Note

The new configuration may not take effect until osn_ip_apply() is called.

3.24.4.8 osn_ip_dns_del()

```
bool osn_ip_dns_del (
             osn_ip_t * ip,
             const osn_ip_addr_t * dns )
```

Remove an DNSv4 server IP from the IPv4 object.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object
in	*dns*	A pointer to a valid DNSv4 address (osn_ip_addr_t)

Note

The new configuration may not take effect until osn_ip_apply() is called.

3.24.4.9 osn_ip_new()

```
osn_ip_t* osn_ip_new (
            const char * ifname )
```

Create a new instance of a IPv4 object.

Parameters

in	*ifname*	Interface name to which the IPv4 instance will be bound to

Returns

This function returns NULL if an error occurs, otherwise a valid osn_ip_t object is returned.

3.24.4.10 osn_ip_route_gw_add()

```
bool osn_ip_route_gw_add (
            osn_ip_t * ip,
            const osn_ip_addr_t * src,
            const osn_ip_addr_t * gw )
```

Add gateway route to IPv4 object.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object
in	*src*	Source IPv4 subnet
in	*gw*	Gateway IPv4 address

Note

The new configuration may not take effect until osn_ip_apply() is called. This might be moved to the OSN_RO↩ UTEV4 API.

3.24.4.11 osn_ip_route_gw_del()

```
bool osn_ip_route_gw_del (
            osn_ip_t * ip,
            const osn_ip_addr_t * src,
            const osn_ip_addr_t * gw )
```

Remove gateway route from IPv4 object.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object
in	*src*	Source IPv4 subnet
in	*gw*	Gateway IPv4 address

Note

The new configuration may not take effect until osn_ip_apply() is called. This might be moved to the OSN_RO↩ UTEV4 API.

3.24.4.12 osn_ip_status_notify()

```
void osn_ip_status_notify (
            osn_ip_t * ip,
            osn_ip_status_fn_t * fn )
```

Set the IPv4 status callback.

Depending on the implementation, the status callback may be invoked periodically or whenever a IPv4 status change has been detected. For maximum portability, the callback implementation should assume it can be called using either mode of operation.

Parameters

in	*ip*	A valid pointer to an osn_ip_t object
in	*fn*	A pointer to the function implementation

3.25 IPv4 Routing

Classes

- struct osn_route_status

Macros

- #define OSN_ROUTE_STATUS_INIT

Typedefs

- typedef struct osn_route osn_route_t
- typedef bool osn_route_status_fn_t(osn_route_t *self, struct osn_route_status *rts, bool remove)

Functions

- osn_route_t * osn_route_new (const char *ifname)
- bool osn_route_del (osn_route_t *self)
- bool osn_route_status_notify (osn_route_t *self, osn_route_status_fn_t *fn)
- void osn_route_data_set (osn_route_t *self, void *data)
- void * osn_route_data_get (osn_route_t *self)

3.25.1 Detailed Description

OpenSync IPv4 Routing API

Note

The IPv4 routing API is subject to change and may be merged with the osn_ip_t class in the future.

3.25.2 Class Documentation

3.25.2.1 struct osn_route_status

Structure passed to the route state notify callback, see osn_route_status_fn_t()

Public Attributes

- osn_ip_addr_t rts_dst_ipaddr
- osn_ip_addr_t rts_dst_mask
- osn_ip_addr_t rts_gw_ipaddr
- osn_mac_addr_t rts_gw_hwaddr

3.25.2.1.1 Member Data Documentation

3.25.2.1.1.1 rts_dst_ipaddr

```
osn_ip_addr_t osn_route_status::rts_dst_ipaddr
```

Destination

3.25.2.1.1.2 rts_dst_mask

```
osn_ip_addr_t osn_route_status::rts_dst_mask
```

Netmask

3.25.2.1.1.3 rts_gw_hwaddr

```
osn_mac_addr_t osn_route_status::rts_gw_hwaddr
```

Gateway MAC address

3.25.2.1.1.4 rts_gw_ipaddr

```
osn_ip_addr_t osn_route_status::rts_gw_ipaddr
```

Gateway, of OSN_IP_ADDR_INIT if none

3.25.3 Macro Definition Documentation

3.25.3.1 OSN_ROUTE_STATUS_INIT

```
#define OSN_ROUTE_STATUS_INIT
```

Value:

```
(struct osn_route_status) \
{                                                   \
    .rts_dst_ipaddr = OSN_IP_ADDR_INIT,             \
    .rts_dst_mask = OSN_IP_ADDR_INIT,               \
    .rts_gw_ipaddr = OSN_IP_ADDR_INIT,              \
    .rts_gw_hwaddr = OSN_MAC_ADDR_INIT,             \
}
```

Initializer for the osn_route_status structure.

Use this macro to initialize a osn_route_status structure to its default values

3.25.4 Typedef Documentation

3.25.4.1 osn_route_status_fn_t

```
typedef bool osn_route_status_fn_t(osn_route_t *self, struct osn_route_status *rts, bool remove)
```

osn_route_t status notification callback. This function will be invoked whenever the osn_route_t object detects a status change and wishes to report it.

Typically this will happen whenever an routing change is detected (for example, when a new route is added to the system).

Some implementation may choose to call this function periodically even if there has been no status change detected.

Parameters

in	*data*	Private data
in	*rts*	A pointer to a osn_route_status
in	*remove*	true if the route in `rts` was removed

3.25.4.2 osn_route_t

```
typedef struct osn_route osn_route_t
```

IPv4 Routing object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_route_new() and must be destroyed using osn_route_del().

3.25.5 Function Documentation

3.25.5.1 osn_route_data_get()

```
void* osn_route_data_get (
            osn_route_t * self )
```

Get user data

Parameters

in	*self*	A valid pointer to an osn_route_t object

3.25.5.2 osn_route_data_set()

```
void osn_route_data_set (
            osn_route_t * self,
            void * data )
```

Set user data

Parameters

in	*self*	A valid pointer to an osn_route_t object
in	*data*	Private data, will be passed to the callback

3.25.5.3 osn_route_del()

```
bool osn_route_del (
            osn_route_t * self )
```

Destroy a valid osn_route_t object.

Parameters

in	*self*	A valid pointer to an osn_route_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

Note

All resources that were allocated during the lifetime of the object are freed.

3.25.5.4 osn_route_new()

```
osn_route_t* osn_route_new (
             const char * ifname )
```

Create a new IPv4 routing object. This object can be used to add/remove IPv4 routing rules. The object is bound to the interface `ifname`.

Parameters

in	*ifname*	Interface name to which the routing object instance will be bound to

Returns

This function returns NULL if an error occurs, otherwise a valid osn_route_t object is returned.

3.25.5.5 osn_route_status_notify()

```
bool osn_route_status_notify (
             osn_route_t * self,
             osn_route_status_fn_t * fn )
```

Set the IPv4 status callback.

Depending on the implementation, the status callback may be invoked periodically or whenever a IPv4 status change has been detected. For maximum portability, the callback implementation should assume it can be called using either mode of operation.

Parameters

in	*self*	A valid pointer to an osn_route_t object
in	*fn*	A pointer to the function implementation

3.26 DHCPv4

Modules

- DHCPv4 Client
- DHCPv4 Server

Macros

- #define OSN_DHCP_FINGERPRINT_MAX 256
- #define OSN_DHCP_VENDORCLASS_MAX 256

Enumerations

- enum osn_notify {
  NOTIFY_UPDATE,
  NOTIFY_DELETE,
  NOTIFY_SYNC,
  NOTIFY_FLUSH }
- enum osn_dhcp_option {
  DHCP_OPTION_SUBNET_MASK = 1,
  DHCP_OPTION_ROUTER = 3,
  DHCP_OPTION_DNS_SERVERS = 6,
  DHCP_OPTION_HOSTNAME = 12,
  DHCP_OPTION_DOMAIN_NAME = 15,
  DHCP_OPTION_BCAST_ADDR = 28,
  DHCP_OPTION_VENDOR_SPECIFIC = 43,
  DHCP_OPTION_ADDRESS_REQUEST = 50,
  DHCP_OPTION_LEASE_TIME = 51,
  DHCP_OPTION_MSG_TYPE = 53,
  DHCP_OPTION_PARAM_LIST = 55,
  DHCP_OPTION_VENDOR_CLASS = 60,
  DHCP_OPTION_DOMAIN_SEARCH = 119,
  DHCP_OPTION_OSYNC_SWVER = 225,
  DHCP_OPTION_OSYNC_PROFILE = 226,
  DHCP_OPTION_OSYNC_SERIAL_OPT = 227,
  DHCP_OPTION_MAX = 256 }

3.26.1 Detailed Description

Common DHCPv4 API definitions

3.26.2 Macro Definition Documentation

3.26.2.1 OSN_DHCP_FINGERPRINT_MAX

```
#define OSN_DHCP_FINGERPRINT_MAX 256
```

Maximum size of a DHCP fingerprint, including the ending \0

3.26.2.2 OSN_DHCP_VENDORCLASS_MAX

```
#define OSN_DHCP_VENDORCLASS_MAX 256
```

Maximum size of a DHCP vendor class string, including the ending \0

3.26.3 Enumeration Type Documentation

3.26.3.1 osn_dhcp_option

```
enum osn_dhcp_option
```

DHCP option list

Note

This list is incomplete. Please find a full list of DHCP options on: https://www.iana.org/assignments/bootp-dhcp-p
xhtml

3.26.3.2 osn_notify

```
enum osn_notify
```

List update protocol

```
 NOTIFY_UPDATE   - Report a new or update a current entry; during SYNC/FLUSH cycled un-flag entry for deletion
 NOTIFY_DELETE   - Delete entry

 NOTIFY_SYNC     - Start synchronization cycle, flag all entries for deletion
 NOTIFY_FLUSH    - Flush all entries flagged for deletion
```

Note

Obsolete.

3.27 DHCPv4 Client

Typedefs

- typedef struct osn_dhcp_client osn_dhcp_client_t
- typedef void osn_dhcp_client_error_fn_t(osn_dhcp_client_t *self)
- typedef bool osn_dhcp_client_opt_notify_fn_t(osn_dhcp_client_t *self, enum osn_notify hint, const char *key, const char *value)

Functions

- osn_dhcp_client_t * osn_dhcp_client_new (const char *ifname)
- bool osn_dhcp_client_del (osn_dhcp_client_t *self)
- bool osn_dhcp_client_start (osn_dhcp_client_t *self)
- bool osn_dhcp_client_stop (osn_dhcp_client_t *self)
- bool osn_dhcp_client_opt_request (osn_dhcp_client_t *self, enum osn_dhcp_option opt, bool request)
- bool osn_dhcp_client_opt_set (osn_dhcp_client_t *self, enum osn_dhcp_option opt, const char *value)
- bool osn_dhcp_client_opt_get (osn_dhcp_client_t *self, enum osn_dhcp_option opt, bool *request, const char **value)
- bool osn_dhcp_client_opt_notify_set (osn_dhcp_client_t *self, osn_dhcp_client_opt_notify_fn_t *fn)
- bool osn_dhcp_client_error_fn_set (osn_dhcp_client_t *self, osn_dhcp_client_error_fn_t *fn)
- bool osn_dhcp_client_vendorclass_set (osn_dhcp_client_t *self, const char *vendorspec)
- bool osn_dhcp_client_state_get (osn_dhcp_client_t *self, bool *enabled)
- void osn_dhcp_client_data_set (osn_dhcp_client_t *self, void *data)
- void * osn_dhcp_client_data_get (osn_dhcp_client_t *self)

3.27.1 Detailed Description

DHCPv4 Client API definitions and functions

3.27.2 Typedef Documentation

3.27.2.1 osn_dhcp_client_error_fn_t

```
typedef void osn_dhcp_client_error_fn_t(osn_dhcp_client_t *self)
```

Error callback function type

3.27.2.2 osn_dhcp_client_opt_notify_fn_t

```
typedef bool osn_dhcp_client_opt_notify_fn_t(osn_dhcp_client_t *self, enum osn_notify hint, const
char *key, const char *value)
```

Notification callback function type

3.27.2.3 osn_dhcp_client_t

```
typedef struct osn_dhcp_client osn_dhcp_client_t
```

OSN DHCPv6 client object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_dhcp_client_new() and must be destroyed using osn_dhcp_client_del().

3.27.3 Function Documentation

3.27.3.1 osn_dhcp_client_data_get()

```
void* osn_dhcp_client_data_get (
            osn_dhcp_client_t * self )
```

Get user data

3.27.3.2 osn_dhcp_client_data_set()

```
void osn_dhcp_client_data_set (
            osn_dhcp_client_t * self,
            void * data )
```

Set user data

3.27.3.3 osn_dhcp_client_del()

```
bool osn_dhcp_client_del (
            osn_dhcp_client_t * self )
```

Destroy a valid osn_dhcpv6_client_t object

3.27.3.4 osn_dhcp_client_error_fn_set()

```
bool osn_dhcp_client_error_fn_set (
            osn_dhcp_client_t * self,
            osn_dhcp_client_error_fn_t * fn )
```

Set the error callback, called whenever an error occurs on the dhcp client (sudden termination or otherwise)

3.27.3.5 osn_dhcp_client_new()

```
osn_dhcp_client_t* osn_dhcp_client_new (
             const char * ifname )
```

Create a new instance of a DHCPv6 client object

3.27.3.6 osn_dhcp_client_opt_get()

```
bool osn_dhcp_client_opt_get (
             osn_dhcp_client_t * self,
             enum osn_dhcp_option opt,
             bool * request,
             const char ** value )
```

Retrieve DHCP option request status and set value (if any)

3.27.3.7 osn_dhcp_client_opt_notify_set()

```
bool osn_dhcp_client_opt_notify_set (
             osn_dhcp_client_t * self,
             osn_dhcp_client_opt_notify_fn_t * fn )
```

Set the option reporting callback, called whenever new DHCP options are received by the DHCP client

3.27.3.8 osn_dhcp_client_opt_request()

```
bool osn_dhcp_client_opt_request (
             osn_dhcp_client_t * self,
             enum osn_dhcp_option opt,
             bool request )
```

Add this option to the server request options, if none is specified a default set will be sent

3.27.3.9 osn_dhcp_client_opt_set()

```
bool osn_dhcp_client_opt_set (
             osn_dhcp_client_t * self,
             enum osn_dhcp_option opt,
             const char * value )
```

Set a DHCP client option – these will be sent to the server

3.27.3.10 osn_dhcp_client_start()

```
bool osn_dhcp_client_start (
             osn_dhcp_client_t * self )
```

Start the DHCP client service

3.27.3.11 osn_dhcp_client_state_get()

```
bool osn_dhcp_client_state_get (
             osn_dhcp_client_t * self,
             bool * enabled )
```

Get the current active state of the DHCP client

3.27.3.12 osn_dhcp_client_stop()

```
bool osn_dhcp_client_stop (
             osn_dhcp_client_t * self )
```

Stop the DHCP client service

3.27.3.13 osn_dhcp_client_vendorclass_set()

```
bool osn_dhcp_client_vendorclass_set (
             osn_dhcp_client_t * self,
             const char * vendorspec )
```

Set the vendor class

3.28 DHCPv4 Server

Classes

- struct osn_dhcp_server_cfg
- struct osn_dhcp_server_lease
- struct osn_dhcp_server_status

Macros

- #define OSN_DHCP_SERVER_CFG_INIT
- #define OSN_DHCP_SERVER_LEASE_INIT

Typedefs

- typedef struct osn_dhcp_server osn_dhcp_server_t
- typedef void osn_dhcp_server_status_fn_t(osn_dhcp_server_t *self, struct osn_dhcp_server_status *status)
- typedef void osn_dhcp_server_error_fn_t(osn_dhcp_server_t *self)

Functions

- osn_dhcp_server_t * osn_dhcp_server_new (const char *ifname)
- bool osn_dhcp_server_del (osn_dhcp_server_t *self)
- void osn_dhcp_server_data_set (osn_dhcp_server_t *self, void *data)
- void * osn_dhcp_server_data_get (osn_dhcp_server_t *self)
- bool osn_dhcp_server_cfg_set (osn_dhcp_server_t *self, struct osn_dhcp_server_cfg *cfg)
- bool osn_dhcp_server_range_add (osn_dhcp_server_t *self, osn_ip_addr_t start, osn_ip_addr_t stop)
- bool osn_dhcp_server_range_del (osn_dhcp_server_t *self, osn_ip_addr_t start, osn_ip_addr_t stop)
- bool osn_dhcp_server_option_set (osn_dhcp_server_t *self, enum osn_dhcp_option opt, const char *value)
- void osn_dhcp_server_error_notify (osn_dhcp_server_t *self, osn_dhcp_server_error_fn_t *fn)
- void osn_dhcp_server_status_notify (osn_dhcp_server_t *self, osn_dhcp_server_status_fn_t *fn)
- bool osn_dhcp_server_reservation_add (osn_dhcp_server_t *self, osn_mac_addr_t macaddr, osn_ip_addr_t ip4addr, const char *hostname)
- bool osn_dhcp_server_reservation_del (osn_dhcp_server_t *self, osn_mac_addr_t macaddr)
- bool osn_dhcp_server_apply (osn_dhcp_server_t *self)

3.28.1 Detailed Description

DHCPv4 Server API definitions and functions

3.28.2 Class Documentation

3.28.2.1 struct osn_dhcp_server_cfg

DHPv4 server configuration parameters. This structure can be used to modify default parameters used by the DHCPv4 server. The structure must be initialized using the OSN_DHCP_SERVER_CFG_INIT initializer. The new configuration can be applied to the DHCPv4 object by calling osn_dhcp_server_cfg_set().

Public Attributes

- int ds_lease_time
- osn_ip_addr_t ds_netmask
- osn_ip_addr_t ds_ipaddr

3.28.2.1.1 Member Data Documentation

3.28.2.1.1.1 ds_ipaddr

```
osn_ip_addr_t osn_dhcp_server_cfg::ds_ipaddr
```

Interface IPv4 address

3.28.2.1.1.2 ds_lease_time

```
int osn_dhcp_server_cfg::ds_lease_time
```

Default lease time in seconds

3.28.2.1.1.3 ds_netmask

```
osn_ip_addr_t osn_dhcp_server_cfg::ds_netmask
```

Interface netmask

3.28.2.2 struct osn_dhcp_server_lease

This structure is used for reporting DHCP lease information. Typically reported as an array inside osn_dhcp_server_↩ status.

Public Attributes

- osn_mac_addr_t dl_hwaddr
- osn_ip_addr_t dl_ipaddr
- char dl_hostname [C_HOSTNAME_LEN]
- char dl_fingerprint [OSN_DHCP_FINGERPRINT_MAX]
- char dl_vendorclass [OSN_DHCP_VENDORCLASS_MAX]
- double dl_leasetime

3.28.2.2.1 Member Data Documentation

3.28.2.2.1.1 dl_fingerprint

```
char osn_dhcp_server_lease::dl_fingerprint[OSN_DHCP_FINGERPRINT_MAX]
```

DHCP fingerprint information

3.28.2.2.1.2 dl_hostname

```
char osn_dhcp_server_lease::dl_hostname[C_HOSTNAME_LEN]
```

Client hostname

3.28.2.2.1.3 dl_hwaddr

```
osn_mac_addr_t osn_dhcp_server_lease::dl_hwaddr
```

Client hardware address

3.28.2.2.1.4 dl_ipaddr

```
osn_ip_addr_t osn_dhcp_server_lease::dl_ipaddr
```

Client IPv4 address

3.28.2.2.1.5 dl_leasetime

```
double osn_dhcp_server_lease::dl_leasetime
```

Lease time in seconds

3.28.2.2.1.6 dl_vendorclass

```
char osn_dhcp_server_lease::dl_vendorclass[OSN_DHCP_VENDORCLASS_MAX]
```

Vendor class information

3.28.2.3 struct osn_dhcp_server_status

DHCPv4 server status structure.

Public Attributes

- char ds_iface
- struct osn_dhcp_server_lease * ds_leases
- int ds_leases_len

3.28.2.3.1 Member Data Documentation

3.28.2.3.1.1 ds_iface

```
char osn_dhcp_server_status::ds_iface
```

Interface name

3.28.2.3.1.2 ds_leases

```
struct osn_dhcp_server_lease* osn_dhcp_server_status::ds_leases
```

Leases array

3.28.2.3.1.3 ds_leases_len

```
int osn_dhcp_server_status::ds_leases_len
```

Leases length

3.28.3 Macro Definition Documentation

3.28.3.1 OSN_DHCP_SERVER_CFG_INIT

```
#define OSN_DHCP_SERVER_CFG_INIT
```

Value:

```
(struct osn_dhcp_server_cfg)    \
{                                                   \
    .ds_lease_time = -1,                            \
    .ds_netmask = OSN_IP_ADDR_INIT,                 \
    .ds_ipaddr = OSN_IP_ADDR_INIT,                  \
}
```

Initializer for the osn_dhcp_server_cfg structure.

3.28.3.2 OSN_DHCP_SERVER_LEASE_INIT

```
#define OSN_DHCP_SERVER_LEASE_INIT
```

Value:

```
(struct osn_dhcp_lease_info) \
{                                                                \
    .dl_hwaddr = OSN_MAC_ADDR_INIT,                              \
    .dl_ipaddr = OSN_IP_ADDR_INIT,                               \
    .dl_leasetime = -1.0,                                        \
}
```

Initializer for osn_dhcp_server_lease. This macro must be used to initialize new instances of struct osn_dhcp_server_lease

3.28.4 Typedef Documentation

3.28.4.1 osn_dhcp_server_error_fn_t

```
typedef void osn_dhcp_server_error_fn_t(osn_dhcp_server_t *self)
```

osn_dhcp_server_t error callback type

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object

3.28.4.2 osn_dhcp_server_status_fn_t

```
typedef void osn_dhcp_server_status_fn_t(osn_dhcp_server_t *self, struct osn_dhcp_server_status
*status)
```

osn_dhcp_server_t status notification callback type

A function of this type, registered via osn_dhcp_server_status_notify, will be invoked whenever the osn_dhcp_server_t object wishes to report the DHCPv4 server status.

Typically this will happen whenever a status change is detected (for example, when a DHCP IP lease has been given out).

Some implementations may choose to call this function periodically even if there has been no status change detected.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*status*	A pointer to a osn_dhcp_server_t status

3.28.4.3 osn_dhcp_server_t

```
typedef struct osn_dhcp_server osn_dhcp_server_t
```

OSN DHCPv4 server object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. An new instance of the object can be obtained by calling osn_dhcp_server_new() and must be destroyed using osn_dhcp_server_del().

3.28.5 Function Documentation

3.28.5.1 osn_dhcp_server_apply()

```
bool osn_dhcp_server_apply (
            osn_dhcp_server_t * self )
```

Ensure that all configuration pertaining the `self` object is applied to the running system.

How the configuration is applied to the system is highly implementation dependent. Sometimes it makes sense to cluster together several configuration parameters (for example, dnsmasq uses a single config file).

osn_dhcp_server_apply() makes sure that a write operation is initiated for all currently cached (dirty) configuration data.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object

Note

It is not guaranteed that the configuration will be applied as soon as osn_dhcp_server_apply() returns – only that the configuration process will be started for all pending operations.

3.28.5.2 osn_dhcp_server_cfg_set()

```
bool osn_dhcp_server_cfg_set (
            osn_dhcp_server_t * self,
            struct osn_dhcp_server_cfg * cfg )
```

Set DHCPv4 server configuration options.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*cfg*	A pointer to DHCPv4 configuration structure

Returns

This function returns true on success. Error is returned in case one or more parameters were found to be invalid or out of range. In such cases the configuration may have been partially applied.

Note

osn_dhcp_server_apply() must be called before this change can take effect.

3.28.5.3 osn_dhcp_server_data_get()

```
void* osn_dhcp_server_data_get (
            osn_dhcp_server_t * self )
```

Get the object `self` private data. If no private data was set, NULL will be returned.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object

Returns

Returns a pointer to private data previously set using osn_dhcp_server_data_set()

3.28.5.4 osn_dhcp_server_data_set()

```
void osn_dhcp_server_data_set (
            osn_dhcp_server_t * self,
            void * data )
```

Set the object `self` private data.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*data*	Pointer to private data

3.28.5.5 osn_dhcp_server_del()

```
bool osn_dhcp_server_del (
            osn_dhcp_server_t * self )
```

Destroy a valid osn_dhcp_server_t object.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

Note

All resources that were allocated during the lifetime of the object are freed.

3.28.5.6 osn_dhcp_server_error_notify()

```
void osn_dhcp_server_error_notify (
            osn_dhcp_server_t * self,
            osn_dhcp_server_error_fn_t * fn )
```

Set the DHCPv4 server error callback.

The error callback is invoked whenever an error condition is detected during run-time (for example, when the server unexpectedly dies).

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*fn*	A pointer to the function implementation

Note

The callback may be invoked only after a successful call to a osn_dhcp_server_apply() function.

3.28.5.7 osn_dhcp_server_new()

```
osn_dhcp_server_t* osn_dhcp_server_new (
            const char * ifname )
```

Create a new instance of a DHCPv4 server object.

Parameters

in	*ifname*	Interface name to which the server instance will be bound to

Returns

This function returns NULL if an error occurs, otherwise a valid osn_dhcp_server_t object is returned.

3.28.5.8 osn_dhcp_server_option_set()

```
bool osn_dhcp_server_option_set (
            osn_dhcp_server_t * self,
            enum osn_dhcp_option opt,
            const char * value )
```

Set a DHCPv4 option value. The option will be sent to the client during the DHCP-OFFER phase.

If an option has already a value assigned, it will be overwritten with the new value.

Using a value of NULL will remove the option from the option list.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*opt*	A DHCPv4 option tag
in	*value*	Option value as string – binary options may require the string to be base64 encoded

Returns

This function returns true on success. False otherwise.

Note

osn_dhcp_server_apply() must be called before this change can take effect.

3.28.5.9 osn_dhcp_server_range_add()

```
bool osn_dhcp_server_range_add (
            osn_dhcp_server_t * self,
            osn_ip_addr_t start,
            osn_ip_addr_t stop )
```

Add a DHCPv4 IP range to the server's lease pool. Many IP ranges can be specified. The caller must make sure the IP ranges do not overlap.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*start*	First IP in range
in	*stop*	Last IP in range

Returns

This function returns true on success. False otherwise.

Note

osn_dhcp_server_apply() must be called before this change can take effect.

3.28.5.10 osn_dhcp_server_range_del()

```
bool osn_dhcp_server_range_del (
            osn_dhcp_server_t * self,
            osn_ip_addr_t start,
            osn_ip_addr_t stop )
```

Remove a DHCPv4 IP range from the server lease pool. An IP range must have been previously registered with osn_dhcp_server_range_del().

It is not possible to slice IP ranges. The `start` and `stop` parameters must match the ones specified to osn_dhcp_↩ server_range_add().

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*start*	First IP in range
in	*stop*	Last IP in range

Returns

This function returns true on success. False otherwise.

Note

osn_dhcp_server_apply() must be called before this change can take effect.

3.28.5.11 osn_dhcp_server_reservation_add()

```
bool osn_dhcp_server_reservation_add (
            osn_dhcp_server_t * self,
            osn_mac_addr_t macaddr,
            osn_ip_addr_t ip4addr,
            const char * hostname )
```

Add a DHCPv4 client IP reservation entry.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*macaddr*	The client MAC address
in	*ip4addr*	The client reserved IP address
in	*hostname*	Desired client hostname - can be NULL if unknown

Returns

This function returns true on success, false otherwise.

Note

osn_dhcp_server_apply() must be called before this change can take effect.

3.28.5.12 osn_dhcp_server_reservation_del()

```
bool osn_dhcp_server_reservation_del (
             osn_dhcp_server_t * self,
             osn_mac_addr_t macaddr )
```

Remove a DHCPv4 IP reservation entry.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*macaddr*	The client MAC address

Returns

This function returns true on success, false otherwise.

Note

osn_dhcp_server_apply() must be called before this change can take effect.

3.28.5.13 osn_dhcp_server_status_notify()

```
void osn_dhcp_server_status_notify (
             osn_dhcp_server_t * self,
             osn_dhcp_server_status_fn_t * fn )
```

Set the DHCPv4 server status callback.

Depending on the implementation, the status callback may be invoked periodically or whenever a DHCP server status change has been detected. For maximum portability, the callback implementation should assume it can be called using either mode of operation.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*fn*	A pointer to the function implementation

3.29 UPnP

Typedefs

- typedef struct osn_upnp osn_upnp_t

Enumerations

- enum osn_upnp_mode {
  UPNP_MODE_NONE,
  UPNP_MODE_INTERNAL,
  UPNP_MODE_EXTERNAL }

Functions

- osn_upnp_t * osn_upnp_new (const char *ifname)
- bool osn_upnp_del (osn_upnp_t *self)
- bool osn_upnp_start (osn_upnp_t *self)
- bool osn_upnp_stop (osn_upnp_t *self)
- bool osn_upnp_set (osn_upnp_t *self, enum osn_upnp_mode mode)
- bool osn_upnp_get (osn_upnp_t *self, enum osn_upnp_mode *mode)

3.29.1 Detailed Description

OpenSync UPnP API

3.29.2 Typedef Documentation

3.29.2.1 osn_upnp_t

```
typedef struct osn_upnp osn_upnp_t
```

OSN UPnP service object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_upnp_new() and must be destroyed using osn_upnp_del().

3.29.3 Enumeration Type Documentation

3.29.3.1 osn_upnp_mode

```
enum osn_upnp_mode
```

UPNP MODE, specifies the mode of the interface:

- UPNP_MODE_NONE - None, this interface is not participating in any UPnP configuration
- UPNP_MODE_INTERNAL - This interface is a UPnP LAN facing interface
- UPNP_MODE_EXTERNAL - This interface is a UPnP WAN facing interface

Enumerator

UPNP_MODE_NONE	Default, no UPnP settings
UPNP_MODE_INTERNAL	This is the LAN facing UPnP interface
UPNP_MODE_EXTERNAL	This is the WAN facing UPnP interface

3.29.4 Function Documentation

3.29.4.1 osn_upnp_del()

```
bool osn_upnp_del (
            osn_upnp_t * self )
```

Destroy a valid osn_upnp_t object.

Parameters

in	*self*	A valid pointer to an osn_upnp_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

3.29.4.2 osn_upnp_get()

```
bool osn_upnp_get (
            osn_upnp_t * self,
            enum osn_upnp_mode * mode )
```

Gets the current UPnP mode on the current interface.

Parameters

in	*self*	A valid pointer to an osn_upnp_t object
out	*mode*	The UPnP mode for this interface as described in osn_upnp_mode

3.29.4.3 osn_upnp_new()

```
osn_upnp_t* osn_upnp_new (
             const char * ifname )
```

Create a new instance of a UPnP object.

Parameters

in	*ifname*	Interface name to which the UPnP instance will be bound

Returns

This function returns NULL if an error occurs, otherwise a valid osn_upnp_t object is returned.

3.29.4.4 osn_upnp_set()

```
bool osn_upnp_set (
             osn_upnp_t * self,
             enum osn_upnp_mode mode )
```

Sets the UPnP mode on the current interface.

Parameters

in	*self*	A valid pointer to an osn_upnp_t object
in	*mode*	The UPnP mode for this interface as described in osn_upnp_mode

3.29.4.5 osn_upnp_start()

```
bool osn_upnp_start (
             osn_upnp_t * self )
```

Start a UPnP server instance

Parameters

in	*self*	A valid pointer to an osn_upnp_t object

Returns

This function returns false if the service was unable to be started.

Note

osn_upnp_start()/osn_upnp_stop() will be replaced with a single call to osn_upnp_apply() in the next release.

3.29.4.6 osn_upnp_stop()

```
bool osn_upnp_stop (
            osn_upnp_t * self )
```

Stop a UPnP server instance

Parameters

in	*self*	A valid pointer to an osn_upnp_t object

Returns

This function returns false if the service was successfully stopped.

Note

osn_upnp_start()/osn_upnp_stop() will be replaced with a single call to osn_upnp_apply() in the next release.

3.30 IPv6

Modules

- Router Advertisement
- DHCPv6

Classes

- struct osn_ip6_neigh
- struct osn_ip6_status

Typedefs

- typedef struct osn_ip6 osn_ip6_t
- typedef void osn_ip6_status_fn_t(osn_ip6_t *self, struct osn_ip6_status *status)

Functions

- osn_ip6_t * osn_ip6_new (const char *ifname)
- bool osn_ip6_del (osn_ip6_t *self)
- bool osn_ip6_apply (osn_ip6_t *self)
- bool osn_ip6_addr_add (osn_ip6_t *self, const osn_ip6_addr_t *addr)
- bool osn_ip6_addr_del (osn_ip6_t *self, const osn_ip6_addr_t *addr)
- bool osn_ip6_dns_add (osn_ip6_t *self, const osn_ip6_addr_t *dns)
- bool osn_ip6_dns_del (osn_ip6_t *self, const osn_ip6_addr_t *dns)
- void osn_ip6_status_notify (osn_ip6_t *self, osn_ip6_status_fn_t *fn)
- void osn_ip6_data_set (osn_ip6_t *self, void *data)
- void * osn_ip6_data_get (osn_ip6_t *self)

3.30.1 Detailed Description

OpenSync IPv6 API

3.30.2 Class Documentation

3.30.2.1 struct osn_ip6_neigh

IPv6 neighbor report structure. Used by osn_ip6_status to report IPv6 neighbors.

Public Attributes

- osn_ip6_addr_t i6n_ipaddr
- osn_mac_addr_t i6n_hwaddr

3.30.2.1.1 Member Data Documentation

3.30.2.1.1.1 i6n_hwaddr

```
osn_mac_addr_t osn_ip6_neigh::i6n_hwaddr
```

Neighbor MAC address

3.30.2.1.1.2 i6n_ipaddr

```
osn_ip6_addr_t osn_ip6_neigh::i6n_ipaddr
```

Neighbor IPv6 address

3.30.2.2 struct osn_ip6_status

IPv6 status structure. A structure of this type is used when reporting the status of the IPv6 object. See osn_ip_status↩ _fn_t() and osn_ip_status_notify() for more details.

Public Attributes

- const char * is6_ifname
- osn_ip6_addr_t * is6_addr
- size_t is6_addr_len
- osn_ip6_addr_t * is6_dns
- size_t is6_dns_len
- struct osn_ip6_neigh * is6_neigh
- size_t is6_neigh_len

3.30.2.2.1 Member Data Documentation

3.30.2.2.1.1 is6_addr

```
osn_ip6_addr_t* osn_ip6_status::is6_addr
```

List of IPv6 addresses on interface

3.30.2.2.1.2 is6_addr_len

```
size_t osn_ip6_status::is6_addr_len
```

Length of is6_addr array

3.30.2.2.1.3 is6_dns

```
osn_ip6_addr_t* osn_ip6_status::is6_dns
```

List of configure DNSv6 servers

3.30.2.2.1.4 is6_dns_len

```
size_t osn_ip6_status::is6_dns_len
```

Length of is6_dns array

3.30.2.2.1.5 is6_ifname

```
const char* osn_ip6_status::is6_ifname
```

Interface name

3.30.2.2.1.6 is6_neigh

```
struct osn_ip6_neigh* osn_ip6_status::is6_neigh
```

List of IPv6 neighbors

3.30.2.2.1.7 is6_neigh_len

```
size_t osn_ip6_status::is6_neigh_len
```

Length of is6_neigh array

3.30.3 Typedef Documentation

3.30.3.1 osn_ip6_status_fn_t

```
typedef void osn_ip6_status_fn_t(osn_ip6_t *self, struct osn_ip6_status *status)
```

osn_ip6_t status notification callback type

A function of this type, registered via osn_ip6_status_notify, will be invoked whenever the osn_ip6_t object wishes to report the IPv6 status.

Typically this will happen whenever an IPv6 status change is detected (for example, when the IP of the interface changes).

Some implementations may choose to call this function periodically even if there has been no status change detected.

Parameters

in	*self*	A valid pointer to an osn_ip6_t object
in	*status*	A pointer to a osn_ip6_status

3.30.3.2 osn_ip6_t

```
typedef struct osn_ip6 osn_ip6_t
```

OSN IPv6 object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_ip6_new() and must be destroyed using osn_ip6_del().

3.30.4 Function Documentation

3.30.4.1 osn_ip6_addr_add()

```
bool osn_ip6_addr_add (
            osn_ip6_t * self,
            const osn_ip6_addr_t * addr )
```

Add an IPv6 address to the IPv6 object.

Parameters

in	*self*	A valid pointer to an osn_ip6_t object
in	*addr*	A pointer to a valid IPv6 address (osn_ip6_addr_t)

Returns

This function returns true if the address was successfully added to the object, false otherwise.

Note

The new configuration may not take effect until osn_ip6_apply() is called.
Adding a duplicate address may result in success.
If osn_ip6_addr_add() returns success when adding a duplicate address then osn_ip6_addr_del() should return success when removing an invalid address.

3.30.4.2 osn_ip6_addr_del()

```
bool osn_ip6_addr_del (
            osn_ip6_t * self,
            const osn_ip6_addr_t * addr )
```

Remove an IPv6 address from the IPv6 object.

Parameters

in	*self*	A valid pointer to an osn_ip6_t object
in	*addr*	A pointer to a valid IPv6 address (osn_ip6_addr_t)

Note

The new configuration may not take effect until osn_ip6_apply() is called.
Removing an non-existent address may result in success.
If osn_ip6_addr_add() returns success when adding a duplicate address then osn_ip6_addr_del() should return success when removing an invalid address.

3.30.4.3 osn_ip6_apply()

```
bool osn_ip6_apply (
            osn_ip6_t * self )
```

Ensure that all configuration pertaining the `self` object is applied to the running system.

How the configuration is applied to the system is highly implementation dependent. Sometimes it makes sense to cluster together several configuration parameters (for example, dnsmasq uses a single config file).

osn_ip6_apply() makes sure that a write operation is initiated for all currently cached (dirty) configuration data.

Note

It is not guaranteed that the configuration will be applied as soon as osn_ip6_apply() returns – only that the configuration process will be started for all pending operations.

3.30.4.4 osn_ip6_data_get()

```
void* osn_ip6_data_get (
            osn_ip6_t * self )
```

Get the IPv6 user data.

Parameters

in	*self*	A valid pointer to an osn_ip6_t object

Returns

This function returns the data that was previously set using osn_ip6_data_set().

3.30.4.5 osn_ip6_data_set()

```
void osn_ip6_data_set (
            osn_ip6_t * self,
            void * data )
```

Set the IPv6 user data.

Parameters

in	*self*	A valid pointer to an osn_ip6_t object
in	*data*	Private data

3.30.4.6 osn_ip6_del()

```
bool osn_ip6_del (
            osn_ip6_t * self )
```

Destroy a valid osn_ip6_t object.

Parameters

in	*self*	A valid pointer to an osn_ip6_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

Note

All global IPv6 addresses that were allocated during the lifetime of the object are removed. The implementation may choose to remove all global IPv6 addresses regardless if they were added using osn_ip6_addr_add().

3.30.4.7 osn_ip6_dns_add()

```
bool osn_ip6_dns_add (
             osn_ip6_t * self,
             const osn_ip6_addr_t * dns )
```

Add an DNSv6 server address to the IPv6 object.

Parameters

in	*self*	A valid pointer to an osn_ip6_t object
in	*dns*	A pointer to a valid IPv6 address (osn_ip6_addr_t)

Note

The new configuration may not take effect until osn_ip6_apply() is called.

3.30.4.8 osn_ip6_dns_del()

```
bool osn_ip6_dns_del (
             osn_ip6_t * self,
             const osn_ip6_addr_t * dns )
```

Remove an DNSv6 server IPv6 address from the IPv6 object.

Parameters

in	*self*	A valid pointer to an osn_ip_t object
in	*dns*	A pointer to a valid DNSv4 address (osn_ip_addr_t)

Note

The new configuration may not take effect until osn_ip_apply() is called.

3.30.4.9 osn_ip6_new()

```
osn_ip6_t* osn_ip6_new (
             const char * ifname )
```

Create a new instance of a IPv6 object.

Parameters

in	*ifname*	Interface name to which the IPv6 instance will be bound to

Returns

This function returns NULL if an error occurs, otherwise a valid osn_ip6_t object is returned.

3.30.4.10 osn_ip6_status_notify()

```
void osn_ip6_status_notify (
            osn_ip6_t * self,
            osn_ip6_status_fn_t * fn )
```

Set the IPv6 status callback.

Depending on the implementation, the status callback may be invoked periodically or whenever a IPv6 status change has been detected. For maximum portability, the callback implementation should assume it can be called using either mode of operation.

Parameters

in	*self*	A valid pointer to an osn_ip6_t object
in	*fn*	A pointer to the function implementation

3.31 Router Advertisement

Classes

- struct osn_ip6_radv_options

Macros

- #define OSN_IP6_RADV_OPTIONS_INIT

Typedefs

- typedef struct osn_ip6_radv osn_ip6_radv_t

Functions

- osn_ip6_radv_t * osn_ip6_radv_new (const char *ifname)
- bool osn_ip6_radv_del (osn_ip6_radv_t *self)
- bool osn_ip6_radv_set (osn_ip6_radv_t *self, const struct osn_ip6_radv_options *opts)
- bool osn_ip6_radv_add_prefix (osn_ip6_radv_t *self, const osn_ip6_addr_t *prefix, bool autonomous, bool on-link)
- bool osn_ip6_radv_del_prefix (osn_ip6_radv_t *self, const osn_ip6_addr_t *prefix)
- bool osn_ip6_radv_add_rdnss (osn_ip6_radv_t *self, const osn_ip6_addr_t *dns)
- bool osn_ip6_radv_del_rdnss (osn_ip6_radv_t *self, const osn_ip6_addr_t *dns)
- bool osn_ip6_radv_add_dnssl (osn_ip6_radv_t *self, char *sl)
- bool osn_ip6_radv_del_dnssl (osn_ip6_radv_t *self, char *sl)
- bool osn_ip6_radv_apply (osn_ip6_radv_t *self)

3.31.1 Detailed Description

OpenSync IPv6 Router Advertisement API

3.31.2 Class Documentation

3.31.2.1 struct osn_ip6_radv_options

Router advertisement options, typically INT_MIN means that the value is unset and the default should be used.

The OSN_IP6_RADV_OPTIONS_INIT macro can be used to initialize the structure to sane defaults.

This structure can be applied to an existing osn_ip6_radv_t object by using the osn_ip6_radv_set() function.

Public Attributes

- bool ra_managed
- bool ra_other_config
- bool ra_home_agent
- int ra_max_adv_interval
- int ra_min_adv_interval
- int ra_default_lft
- int ra_preferred_router
- int ra_mtu
- int ra_reachable_time
- int ra_retrans_timer
- int ra_current_hop_limit

3.31.2.1.1 Member Data Documentation

3.31.2.1.1.1 ra_current_hop_limit

```
int osn_ip6_radv_options::ra_current_hop_limit
```

Current hop limit

3.31.2.1.1.2 ra_default_lft

```
int osn_ip6_radv_options::ra_default_lft
```

Default lifetime

3.31.2.1.1.3 ra_home_agent

```
bool osn_ip6_radv_options::ra_home_agent
```

Home Agent flag

3.31.2.1.1.4 ra_managed

```
bool osn_ip6_radv_options::ra_managed
```

Managed flag

3.31.2.1.1.5 ra_max_adv_interval

```
int osn_ip6_radv_options::ra_max_adv_interval
```

Max advertisement interval

3.31.2.1.1.6 ra_min_adv_interval

```
int osn_ip6_radv_options::ra_min_adv_interval
```

Min advertisement interval

3.31.2.1.1.7 ra_mtu

```
int osn_ip6_radv_options::ra_mtu
```

Advertised MTU

3.31.2.1.1.8 ra_other_config

```
bool osn_ip6_radv_options::ra_other_config
```

Other Config flag

3.31.2.1.1.9 ra_preferred_router

```
int osn_ip6_radv_options::ra_preferred_router
```

Preferred router: 0 - low, 1 - med, 2 - high

3.31.2.1.1.10 ra_reachable_time

```
int osn_ip6_radv_options::ra_reachable_time
```

Reachable time

3.31.2.1.1.11 ra_retrans_timer

```
int osn_ip6_radv_options::ra_retrans_timer
```

Retransmit timer

3.31.3 Macro Definition Documentation

3.31.3.1 OSN_IP6_RADV_OPTIONS_INIT

```
#define OSN_IP6_RADV_OPTIONS_INIT
```

Value:

```
(struct osn_ip6_radv_options) \
{                                                                   \
    .ra_max_adv_interval    = INT_MIN,                              \
    .ra_min_adv_interval    = INT_MIN,                              \
    .ra_default_lft         = INT_MIN,                              \
    .ra_preferred_router    = INT_MIN,                              \
    .ra_mtu                 = INT_MIN,                              \
    .ra_reachable_time      = INT_MIN,                              \
    .ra_retrans_timer       = INT_MIN,                              \
    .ra_current_hop_limit   = INT_MIN,                              \
}
```

Initialization helper for struct osn_ip6_radv_options

3.31.4 Typedef Documentation

3.31.4.1 osn_ip6_radv_t

```
typedef struct osn_ip6_radv osn_ip6_radv_t
```

IPv6 Router Advertisement object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_ip6_radv_new() and must be destroyed using osn_ip6_radv_del().

3.31.5 Function Documentation

3.31.5.1 osn_ip6_radv_add_dnssl()

```
bool osn_ip6_radv_add_dnssl (
            osn_ip6_radv_t * self,
            char * sl )
```

Add the DNSv6 domain to the list of advertised DNSSL domains

Parameters

in	*self*	A valid pointer to an osn_ip6_radv_t object
in	*sl*	Search domain

Returns

Return true if the DNSv6search domain server address was successfully added, false otherwise.

Note

If osn_ip6_radv_add_dnssl() allows the addition of duplicate domains, then osn_ip6_radv_del_dnssl() should allow removal of non-existing domains.
The new configuration may not take effect until osn_ip6_radv_apply() is called.

3.31.5.2 osn_ip6_radv_add_prefix()

```
bool osn_ip6_radv_add_prefix (
            osn_ip6_radv_t * self,
            const osn_ip6_addr_t * prefix,
            bool autonomous,
            bool onlink )
```

Add the IPv6 prefix to the list of advertised prefixes.

Parameters

in	*self*	A valid pointer to an osn_ip6_radv_t object
in	*prefix*	IPv6 prefix to add
in	*autonomous*	Set the A flag for this prefix
in	*onlink*	Set the L flag for this prefix

Returns

Return true if the prefix was successfully added, false otherwise.

Note

If osn_ip6_radv_add_prefix() allows the addition of duplicate prefixes, then osn_ip6_radv_del_prefix() should allow removal of non-existing prefixes.
The new configuration may not take effect until osn_ip6_radv_apply() is called.

3.31.5.3 osn_ip6_radv_add_rdnss()

```
bool osn_ip6_radv_add_rdnss (
            osn_ip6_radv_t * self,
            const osn_ip6_addr_t * dns )
```

Add the IPv6 address to the list of advertised RDNS servers.

Parameters

in	*self*	A valid pointer to an osn_ip6_radv_t object
in	*dns*	IPv6 DNS server address to add

Returns

Return true if the DNSv6 server address was successfully added, false otherwise.

Note

If osn_ip6_radv_add_rdnss() allows the addition of duplicate addresses, then osn_ip6_radv_del_rdnss() should allow removal of non-existing addresses.
The new configuration may not take effect until osn_ip6_radv_apply() is called.

3.31.5.4 osn_ip6_radv_apply()

```
bool osn_ip6_radv_apply (
            osn_ip6_radv_t * self )
```

Applies previously configured settings

Parameters

in	*self*	A valid pointer to an osn_ip6_radv_t object

Returns

This function returns true on success. On error, false is returned.

3.31.5.5 osn_ip6_radv_del()

```
bool osn_ip6_radv_del (
            osn_ip6_radv_t * self )
```

Destroys the a Router Advertisement object. This function should return the system to its original state (stopping any RADV services that are running as a consequence of this object, etc.)

Parameters

in	*self*	A valid pointer to an osn_ip6_radv_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

3.31.5.6 osn_ip6_radv_del_dnssl()

```
bool osn_ip6_radv_del_dnssl (
            osn_ip6_radv_t * self,
            char * sl )
```

Remove the DNSv6 search domain from the list of advertised DNSSL servers.

Parameters

in	*self*	A valid pointer to an osn_ip6_radv_t object
in	*sl*	IPv6 DNS server address to remove

Returns

Return true if the DNSv6 search domain server address was successfully removed, false otherwise.

Note

If osn_ip6_radv_add_dnssl() allows the addition of duplicate domains, then osn_ip6_radv_del_dnssl() should allow removal of non-existing domains.
The new configuration may not take effect until osn_ip6_radv_apply() is called.

3.31.5.7 osn_ip6_radv_del_prefix()

```
bool osn_ip6_radv_del_prefix (
            osn_ip6_radv_t * self,
            const osn_ip6_addr_t * prefix )
```

Remove the IPv6 prefix from the list of advertised prefixes.

Parameters

in	*self*	A valid pointer to an osn_ip6_radv_t object
in	*prefix*	IPv6 prefix to remove

Returns

Return true if the prefix was successfully added, false otherwise.

Note

If osn_ip6_radv_add_prefix() allows the addition of duplicate prefixes, then osn_ip6_radv_del_prefix() should allow removal of non-existing prefixes.
The new configuration may not take effect until osn_ip6_radv_apply() is called.

3.31.5.8 osn_ip6_radv_del_rdnss()

```
bool osn_ip6_radv_del_rdnss (
            osn_ip6_radv_t * self,
            const osn_ip6_addr_t * dns )
```

Remove the IPv6 address from the list of advertised RDNS servers.

Parameters

in	*self*	A valid pointer to an osn_ip6_radv_t object
in	*dns*	IPv6 DNS server address to remove

Returns

Return true if the DNSv6 server address was successfully added, false otherwise.

Note

If osn_ip6_radv_add_rdnss() allows the addition of duplicate addresses, then osn_ip6_radv_del_rdnss() should allow removal of non-existing addresses.
The new configuration may not take effect until osn_ip6_radv_apply() is called.

3.31.5.9 osn_ip6_radv_new()

```
osn_ip6_radv_t* osn_ip6_radv_new (
            const char * ifname )
```

Create a new instance of the Router Advertisement object.

Parameters

in	*ifname*	Interface name to which the IPv4 instance will be bound to

Returns

This function returns NULL if an error occurs, otherwise a valid osn_ip6_radv_t object is returned.

3.31.5.10 osn_ip6_radv_set()

```
bool osn_ip6_radv_set (
            osn_ip6_radv_t * self,
            const struct osn_ip6_radv_options * opts )
```

Apply the Router Advertisement options to the osn_ip6_radv_t object. The osn_ip6_radv_options can be used to modify the default parameters of the RA service. The osn_ip6_radv_options structure must be initialized with the OSN_IP6_RADV_OPTIONS_INIT initializer.

Parameters

in	*self*	A valid pointer to an osn_ip6_radv_t object
in	*opts*	A valid pointer to an osn_ip6_radv_options structure

Returns

This function returns true on success, false otherwise. If false is returned options may have been partially applied.

Note

The new configuration may not take effect until osn_ip6_radv_apply() is called.

3.32 DHCPv6

Modules

- DHCPv6 Client
- DHCPv6 Server

Macros

- #define OSN_DHCP_HOSTNAME_LEN 64
- #define OSN_DHCP_OPTIONS_MAX 256

3.32.1 Detailed Description

Common DHCPv6 API definition

3.32.2 Macro Definition Documentation

3.32.2.1 OSN_DHCP_HOSTNAME_LEN

```
#define OSN_DHCP_HOSTNAME_LEN 64
```

Maximum hostname size including the terminating \0

3.32.2.2 OSN_DHCP_OPTIONS_MAX

```
#define OSN_DHCP_OPTIONS_MAX 256
```

Maximum number of DHCP options

3.33 DHCPv6 Client

Classes

- struct osn_dhcpv6_client_status

Typedefs

- typedef struct osn_dhcpv6_client osn_dhcpv6_client_t
- typedef void osn_dhcpv6_client_status_fn_t(osn_dhcpv6_client_t *self, struct osn_dhcpv6_client_status *status)

Functions

- osn_dhcpv6_client_t * osn_dhcpv6_client_new (const char *ifname)
- bool osn_dhcpv6_client_del (osn_dhcpv6_client_t *self)
- bool osn_dhcpv6_client_set (osn_dhcpv6_client_t *self, bool request_address, bool request_prefixes, bool rapid_commit, bool renew)
- bool osn_dhcpv6_client_option_request (osn_dhcpv6_client_t *self, int tag)
- bool osn_dhcpv6_client_option_send (osn_dhcpv6_client_t *self, int tag, const char *value)
- void osn_dhcpv6_client_status_notify (osn_dhcpv6_client_t *self, osn_dhcpv6_client_status_fn_t *fn)
- bool osn_dhcpv6_client_apply (osn_dhcpv6_client_t *self)
- void osn_dhcpv6_client_data_set (osn_dhcpv6_client_t *self, void *data)
- void * osn_dhcpv6_client_data_get (osn_dhcpv6_client_t *self)

3.33.1 Detailed Description

DHCPv6 Client API definition

3.33.2 Class Documentation

3.33.2.1 struct osn_dhcpv6_client_status

DHCPv6 client status report structure. A structure of this type is used for reporting the status of the DHCPv6 client object. See osn_dhcpv6_client_status_fn_t

Public Attributes

- bool d6c_connected
- char * d6c_recv_options [OSN_DHCP_OPTIONS_MAX]

3.33.2.1.1 Member Data Documentation

3.33.2.1.1.1 d6c_connected

```
bool osn_dhcpv6_client_status::d6c_connected
```

True whether client has connected

3.33.2.1.1.2 d6c_recv_options

```
char* osn_dhcpv6_client_status::d6c_recv_options[OSN_DHCP_OPTIONS_MAX]
```

Received options, base64 encoded string or NULL for none

3.33.3 Typedef Documentation

3.33.3.1 osn_dhcpv6_client_status_fn_t

```
typedef void osn_dhcpv6_client_status_fn_t(osn_dhcpv6_client_t *self, struct osn_dhcpv6_client_↩
status *status)
```

osn_dhcpv6_client_t status notification callback type

A function of this type, registered via osn_dhcpv6_client_status_notify, will be invoked whenever the osn_dhcpv6_↩ client_t object wishes to report the DHCPv6 client status.

Typically this will happen whenever a status change is detected (for example, when DHCPv6 client options are received).

Some implementations may choose to call this function periodically even if there has been no status change detected.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_client_t object
in	*status*	A pointer to a osn_dhcpv6_client_status

3.33.3.2 osn_dhcpv6_client_t

```
typedef struct osn_dhcpv6_client osn_dhcpv6_client_t
```

OSN DHCPv6 client object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_dhcpv6_client_new() and must be destroyed using osn_dhcpv6_client_del().

3.33.4 Function Documentation

3.33.4.1 osn_dhcpv6_client_apply()

```
bool osn_dhcpv6_client_apply (
            osn_dhcpv6_client_t * self )
```

Ensure that all configuration pertaining the `self` object is applied to the running system.

How the configuration is applied to the system is highly implementation dependent. Sometimes it makes sense to cluster together several configuration parameters (for example, dnsmasq uses a single config file).

The osn_Dhcpv6_client_apply() function typically restarts the DHCPv6 client.

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object

3.33.4.2 osn_dhcpv6_client_data_get()

```
void* osn_dhcpv6_client_data_get (
            osn_dhcpv6_client_t * self )
```

Get user data associated with object `self`

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object

3.33.4.3 osn_dhcpv6_client_data_set()

```
void osn_dhcpv6_client_data_set (
            osn_dhcpv6_client_t * self,
            void * data )
```

Set user data associated with object `self`

Parameters

in	*self*	A valid pointer to an osn_dhcp_server_t object
in	*data*	User data

3.33.4.4 osn_dhcpv6_client_del()

```
bool osn_dhcpv6_client_del (
            osn_dhcpv6_client_t * self )
```

Destroy a valid osn_dhcpv6_client_t object.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_client_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

3.33.4.5 osn_dhcpv6_client_new()

```
osn_dhcpv6_client_t* osn_dhcpv6_client_new (
            const char * ifname )
```

Create a new instance of a DHCPv6 client object.

Parameters

in	*ifname*	Interface name to which the DHCPv6 client instance will be bound

Returns

This function returns NULL if an error occurs, otherwise a valid osn_dhcpv6_client_t object is returned.

3.33.4.6 osn_dhcpv6_client_option_request()

```
bool osn_dhcpv6_client_option_request (
            osn_dhcpv6_client_t * self,
            int tag )
```

Set various options that will be requested from the server during the DHCP_REQUEST phase.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_client_t object
in	*tag*	The DHCPv6 option id

Returns

This function returns true if the option was successfully set, false otherwise.

Note

There's currently no API to unset a requested option.

3.33.4.7 osn_dhcpv6_client_option_send()

```
bool osn_dhcpv6_client_option_send (
            osn_dhcpv6_client_t * self,
            int tag,
            const char * value )
```

Set various DHCPv6 options that will be sent to the DHCPv6 server during the DHCP_REQUEST phase.

A value of NULL indicates that the options should be removed from the request list.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_client_t object
in	*tag*	The DHCPv6 option id
in	*value*	DHCPv6 option value or NULL

Returns

This function returns true if the option was successfully set, false otherwise.

3.33.4.8 osn_dhcpv6_client_set()

```
bool osn_dhcpv6_client_set (
            osn_dhcpv6_client_t * self,
            bool request_address,
            bool request_prefixes,
            bool rapid_commit,
            bool renew )
```

Set DHCPv6 client options.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_client_t object
in	*request_address*	Request a DHCPv6 address
in	*request_prefixes*	Request DHCPv6 prefixes
in	*rapid_commit*	use fast rapid commit
in	*renew*	renew the IPv6 address

Returns

This function returns true on success or false on error.

Note

If an error is returned, the options may be partially applied.

3.33.4.9 osn_dhcpv6_client_status_notify()

```
void osn_dhcpv6_client_status_notify (
            osn_dhcpv6_client_t * self,
            osn_dhcpv6_client_status_fn_t * fn )
```

Set the DHCPv6 client status callback.

Depending on the implementation, the status callback may be invoked periodically or whenever the DHCPv6 client status change has been detected (for example, when new DHCPv6 options are received). For maximum portability, the callback implementation should assume it can be called using either mode of operation.

Parameters

in	*self*	A valid pointer to an osn_ip6_t object
in	*fn*	A pointer to the function implementation

3.34 DHCPv6 Server

Classes

- struct osn_dhcpv6_server_prefix
- struct osn_dhcpv6_server_lease
- struct osn_dhcpv6_server_status

Typedefs

- typedef struct osn_dhcpv6_server osn_dhcpv6_server_t
- typedef void osn_dhcpv6_server_status_fn_t(osn_dhcpv6_server_t *d6s, struct osn_dhcpv6_server_status *status)

Functions

- osn_dhcpv6_server_t * osn_dhcpv6_server_new (const char *iface)
- bool osn_dhcpv6_server_del (osn_dhcpv6_server_t *self)
- bool osn_dhcpv6_server_apply (osn_dhcpv6_server_t *self)
- void osn_dhcpv6_server_data_set (osn_dhcpv6_server_t *self, void *data)
- void * osn_dhcpv6_server_data_get (osn_dhcpv6_server_t *self)
- bool osn_dhcpv6_server_prefix_add (osn_dhcpv6_server_t *self, struct osn_dhcpv6_server_prefix *prefix)
- bool osn_dhcpv6_server_prefix_del (osn_dhcpv6_server_t *self, struct osn_dhcpv6_server_prefix *prefix)
- bool osn_dhcpv6_server_option_send (osn_dhcpv6_server_t *self, int tag, const char *value)
- bool osn_dhcpv6_server_lease_add (osn_dhcpv6_server_t *self, struct osn_dhcpv6_server_lease *lease)
- bool osn_dhcpv6_server_lease_del (osn_dhcpv6_server_t *self, struct osn_dhcpv6_server_lease *lease)
- bool osn_dhcpv6_server_status_notify (osn_dhcpv6_server_t *self, osn_dhcpv6_server_status_fn_t *fn)

3.34.1 Detailed Description

DHCPv6 Server API definition

3.34.2 Class Documentation

3.34.2.1 struct osn_dhcpv6_server_prefix

DHCPv6 prefix. This structure is used to add a prefix range to a DHCPv6 server configuration. It can be configured using the osn_dhcpv6_server_prefix_add() function and can be removed by using the osn_dhcpv6_server_prefix_del() function.

Public Attributes

- osn_ip6_addr_t d6s_prefix
- bool ds6_onlink
- bool ds6_autonomous

3.34.2.1.1 Member Data Documentation

3.34.2.1.1.1 d6s_prefix

```
osn_ip6_addr_t osn_dhcpv6_server_prefix::d6s_prefix
```

The DHCPv6 server prefix

3.34.2.1.1.2 ds6_autonomous

```
bool osn_dhcpv6_server_prefix::ds6_autonomous
```

Unused

3.34.2.1.1.3 ds6_onlink

```
bool osn_dhcpv6_server_prefix::ds6_onlink
```

Onlink flag of prefix

3.34.2.2 struct osn_dhcpv6_server_lease

This structure is used to add static DHCPv6 leases. It must be added using the osn_dhcpv6_server_lease_add() function and can be deleted using using osn_dhcpv6_server_lease_del() function.

Public Attributes

- osn_ip6_addr_t d6s_addr
- char d6s_duid [261]
- osn_mac_addr_t d6s_hwaddr
- char d6s_hostname [OSN_DHCP_HOSTNAME_LEN]
- int d6s_leased_time

3.34.2.2.1 Member Data Documentation

3.34.2.2.1.1 d6s_addr

```
osn_ip6_addr_t osn_dhcpv6_server_lease::d6s_addr
```

IPV6 address

3.34.2.2.1.2 d6s_duid

```
char osn_dhcpv6_server_lease::d6s_duid[261]
```

Client DUID

3.34.2.2.1.3 d6s_hostname

```
char osn_dhcpv6_server_lease::d6s_hostname[OSN_DHCP_HOSTNAME_LEN]
```

Hostname

3.34.2.2.1.4 d6s_hwaddr

```
osn_mac_addr_t osn_dhcpv6_server_lease::d6s_hwaddr
```

Hardware address

3.34.2.2.1.5 d6s_leased_time

```
int osn_dhcpv6_server_lease::d6s_leased_time
```

Leased time

3.34.2.3 struct osn_dhcpv6_server_status

DHCPv6 server status report structure. A structure of this type is used for reporting the status of the DHCPv6 server object. See osn_dhcpv6_server_status_fn_t

Public Attributes

- char * d6st_iface
- int d6st_leases_len
- struct osn_dhcpv6_server_lease * d6st_leases

3.34.2.3.1 Member Data Documentation

3.34.2.3.1.1 d6st_iface

```
char* osn_dhcpv6_server_status::d6st_iface
```

Interface of DHCPv6 server instance

3.34.2.3.1.2 d6st_leases

```
struct osn_dhcpv6_server_lease* osn_dhcpv6_server_status::d6st_leases
```

Currently active leases

3.34.2.3.1.3 d6st_leases_len

```
int osn_dhcpv6_server_status::d6st_leases_len
```

Number of active leases

3.34.3 Typedef Documentation

3.34.3.1 osn_dhcpv6_server_status_fn_t

```
typedef void osn_dhcpv6_server_status_fn_t(osn_dhcpv6_server_t *d6s, struct osn_dhcpv6_server_↩
status *status)
```

DHCPv6 server status reporting callback type

A function of this type, registered via osn_dhcpv6_server_status_notify() will receive status events from the DHCPv6 server object. The callback may be invoked before the osn_dhcpv6_server_apply() function is called. This can be used to report the DHCPv6 status without applying any system configuration.

Parameters

in	*d6s*	DHPCv6 server object
in	*status*	A structure of type osn_dhcpv6_server_status

3.34.3.2 osn_dhcpv6_server_t

```
typedef struct osn_dhcpv6_server osn_dhcpv6_server_t
```

OSN DHCPv6 server object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_dhcpv6_server_new() and must be destroyed using osn_dhcpv6_server_del().

3.34.4 Function Documentation

3.34.4.1 osn_dhcpv6_server_apply()

```
bool osn_dhcpv6_server_apply (
            osn_dhcpv6_server_t * self )
```

Ensure that all configuration pertaining the `self` object is applied to the running system.

How the configuration is applied to the system is highly implementation dependent. Sometimes it makes sense to cluster together several configuration parameters (for example, dnsmasq uses a single config file).

The osn_dhcpv6_server_apply() function typically restarts the DHCPv6 server.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object

Returns

This function returns true when the configuration was successfully applied to the system, false otherwise.

3.34.4.2 osn_dhcpv6_server_data_get()

```
void* osn_dhcpv6_server_data_get (
            osn_dhcpv6_server_t * self )
```

Retrieve the custom data associated with the object.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object

Returns

This function returns the pointer which was previously set with osn_dhcpv6_server_data_set() or NULL if no data was set.

3.34.4.3 osn_dhcpv6_server_data_set()

```
void osn_dhcpv6_server_data_set (
            osn_dhcpv6_server_t * self,
            void * data )
```

Set the DHCPv6 server object private data. The data can be used to set custom data that is associated with the object.

The private data can be retrieved with osn_dhcpv6_server_data_get()

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object
in	*data*	an opaque pointer to custom data

3.34.4.4 osn_dhcpv6_server_del()

```
bool osn_dhcpv6_server_del (
            osn_dhcpv6_server_t * self )
```

Destroy a valid osn_dhcpv6_server_t object.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

3.34.4.5 osn_dhcpv6_server_lease_add()

```
bool osn_dhcpv6_server_lease_add (
            osn_dhcpv6_server_t * self,
            struct osn_dhcpv6_server_lease * lease )
```

Add a DHCPv6 client lease to the DHCPv6 server configuration. The new configuration must not take effect until osn_dhcpv6_server_apply() is called.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object
in	*lease*	A pointer to a structure of type osn_dhcpv6_server_lease

Returns

This function returns true on success, false otherwise. If this function returns success when removing a non-existing entry, then osn_dhcpv6_server_lease_add() must also return success adding a duplicate entry.

3.34.4.6 osn_dhcpv6_server_lease_del()

```
bool osn_dhcpv6_server_lease_del (
             osn_dhcpv6_server_t * self,
             struct osn_dhcpv6_server_lease * lease )
```

Remove an DHCPv6 client lease from the DHCPv6 server configuration. The new configuration must not take effect until osn_dhcpv6_server_apply() is called.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object
in	*lease*	A pointer to a structure of type osn_dhcpv6_server_lease. This structure defines the entry that will be removed.

Returns

This function returns true on success, false otherwise. If this function returns success when removing a non-existing entry, then osn_dhcpv6_server_lease_add() must also return success adding a duplicate entry.

3.34.4.7 osn_dhcpv6_server_new()

```
osn_dhcpv6_server_t* osn_dhcpv6_server_new (
             const char * iface )
```

Create a new instance of a DHCPv6 server object.

Parameters

in	*iface*	Interface name to which the DHCPv6 server instance will be bound

Returns

This function returns NULL if an error occurs, otherwise a valid osn_dhcpv6_server_t object is returned.

3.34.4.8 osn_dhcpv6_server_option_send()

```
bool osn_dhcpv6_server_option_send (
            osn_dhcpv6_server_t * self,
            int tag,
            const char * value )
```

Add option with ID `tag` to the list of options that the DHCPv6 server will be sending to DHCPv6 clients. The configuration may take effect only after osn_dhcpv6_server_apply() is called.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object
in	*tag*	DHCPv6 option id
in	*value*	DHCPv6 option value or NULL to remove the option

Returns

This function returns true if the option was successfully removed or added to the DHCPv6 configuration.

3.34.4.9 osn_dhcpv6_server_prefix_add()

```
bool osn_dhcpv6_server_prefix_add (
            osn_dhcpv6_server_t * self,
            struct osn_dhcpv6_server_prefix * prefix )
```

Add a DHCPv6 server prefix to the DHCPv6 server configuration. The new configuration must not take effect until osn_dhcpv6_server_apply() is called.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object
in	*prefix*	A pointer to a structure of type osn_dhcpv6_server_prefix

Returns

This function returns true on success, false otherwise. If this function returns success when adding an existing entry, then osn_dhcpv6_server_prefix_del() must also return success when removing a non-existing entry.

3.34.4.10 osn_dhcpv6_server_prefix_del()

```
bool osn_dhcpv6_server_prefix_del (
            osn_dhcpv6_server_t * self,
            struct osn_dhcpv6_server_prefix * prefix )
```

Remove a DHCPv6 server prefix from the DHCPv6 server configuration. The new configuration must not take effect until osn_dhcpv6_server_apply() is called.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object
in	*prefix*	A pointer to a structure of type osn_dhcpv6_server_prefix

Note

Only the prefix->d6s_prefix field is used when looking up entries to be removed.

Returns

This function returns true on success, false otherwise. If this function returns success when removing a non-existing entry, then osn_dhcpv6_server_prefix_add() must also return success adding a duplicate entry.

3.34.4.11 osn_dhcpv6_server_status_notify()

```
bool osn_dhcpv6_server_status_notify (
            osn_dhcpv6_server_t * self,
            osn_dhcpv6_server_status_fn_t * fn )
```

Register a DHCPv6 server status notification callback. If `fn` is NULL, the previous callback is unregistered.

Parameters

in	*self*	A valid pointer to an osn_dhcpv6_server_t object
in	*fn*	A function of type osn_dhcpv6_server_status_fn_t

Returns

This function returns true if the callback was successfully registered, false otherwise.

3.35 L2 Interface

Modules

- Ethernet Interface

3.35.1 Detailed Description

OpenSync L2 Interface Management API

3.36 Ethernet Interface

Classes

- struct osn_netif_status

Typedefs

- typedef struct osn_netif osn_netif_t
- typedef void osn_netif_status_fn_t(osn_netif_t *self, struct osn_netif_status *status)

Functions

- osn_netif_t * osn_netif_new (const char *ifname)
- bool osn_netif_del (osn_netif_t *self)
- void osn_netif_data_set (osn_netif_t *self, void *data)
- void * osn_netif_data_get (osn_netif_t *self)
- void osn_netif_status_notify (osn_netif_t *self, osn_netif_status_fn_t *fn)
- bool osn_netif_apply (osn_netif_t *self)
- bool osn_netif_state_set (osn_netif_t *self, bool up)
- bool osn_netif_mtu_set (osn_netif_t *self, int mtu)
- bool osn_netif_hwaddr_set (osn_netif_t *self, osn_mac_addr_t hwaddr)

3.36.1 Detailed Description

Ethernet Interface (L2) API

3.36.2 Class Documentation

3.36.2.1 struct osn_netif_status

Network interface status structure. A structure of this type is used when reporting the status of the network interface. See osn_netif_status_fn_t() and osn_netif_status_notify() for more details.

Note

If the ns_exists field is false, all subsequent fields should be considered undefined.

Public Attributes

- const char * ns_ifname
- bool ns_exists
- osn_mac_addr_t ns_hwaddr
- bool ns_up
- bool ns_carrier
- int ns_mtu

3.36.2.1.1 Member Data Documentation

3.36.2.1.1.1 ns_carrier

```
bool osn_netif_status::ns_carrier
```

True if carrier was detected (RUNNING)

3.36.2.1.1.2 ns_exists

```
bool osn_netif_status::ns_exists
```

True if interface exists – Subsequent fields should be considered undefined if this is false

3.36.2.1.1.3 ns_hwaddr

```
osn_mac_addr_t osn_netif_status::ns_hwaddr
```

Interface hardware address

3.36.2.1.1.4 ns_ifname

```
const char* osn_netif_status::ns_ifname
```

Interface name

3.36.2.1.1.5 ns_mtu

```
int osn_netif_status::ns_mtu
```

MTU

3.36.2.1.1.6 ns_up

```
bool osn_netif_status::ns_up
```

True if interface is UP

3.36.3 Typedef Documentation

3.36.3.1 osn_netif_status_fn_t

```
typedef void osn_netif_status_fn_t(osn_netif_t *self, struct osn_netif_status *status)
```

osn_netif_t status notification callback type

A function of this type, registered via osn_netif_status_notify, will be invoked whenever the osn_netif_t object wishes to report the status of the network interface.

Typically this will happen whenever a status change is detected (for example, when carrier is detected).

Some implementations may choose to call this function periodically even if there has been no status change detected.

Parameters

in	*self*	The object that is reporting the status
in	*status*	A pointer to a osn_netif_status structure

3.36.3.2 osn_netif_t

```
typedef struct osn_netif osn_netif_t
```

OSN NETIF object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_netif_new() and must be destroyed using osn_netif_del().

3.36.4 Function Documentation

3.36.4.1 osn_netif_apply()

```
bool osn_netif_apply (
            osn_netif_t * self )
```

Ensure that all configuration pertaining the `self` object is applied to the running system.

How the configuration is applied to the system is highly implementation dependent. Sometimes it makes sense to cluster together several configuration parameters.

osn_netif_apply() makes sure that a write operation is initiated for all currently cached (dirty) configuration data.

Note

It is not guaranteed that the configuration will be applied as soon as osn_netif_apply() returns – only that the configuration process will be started for all pending operations.

3.36.4.2 osn_netif_data_get()

```
void* osn_netif_data_get (
            osn_netif_t * self )
```

Get the object `self` user data. If no user data was set, NULL will be returned.

Parameters

in	*self*	A valid pointer to an osn_netif_t object

Returns

Returns a pointer to user data previously set using osn_netif_data_set().

3.36.4.3 osn_netif_data_set()

```
void osn_netif_data_set (
            osn_netif_t * self,
            void * data )
```

Set the object `self` user data.

Parameters

in	*self*	A valid pointer to an osn_netif_t object
in	*data*	Pointer to user data

3.36.4.4 osn_netif_del()

```
bool osn_netif_del (
            osn_netif_t * self )
```

Destroy a valid osn_netif_t object.

Parameters

in	*self*	A valid pointer to an osn_netif_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

Note

All resources that were allocated during the lifetime of the object are freed.

3.36.4.5 osn_netif_hwaddr_set()

```
bool osn_netif_hwaddr_set (
            osn_netif_t * self,
            osn_mac_addr_t hwaddr )
```

Set the interface hardware address.

Parameters

in	*self*	A valid pointer to an osn_eth_t object
in	*hwaddr*	The new interface hardware address

Returns

Returns true if the hardware address was successfully set.

Note

A call to osn_netif_apply() may be required before the change can take effect.

3.36.4.6 osn_netif_mtu_set()

```
bool osn_netif_mtu_set (
            osn_netif_t * self,
            int mtu )
```

Set the interface MTU.

Parameters

in	*self*	A valid pointer to an osn_netif_t object
in	*mtu*	New MTU value

Returns

Returns true if the MTU was successfully set. Failing a valid range check may result in a return status of false.

Note

A call to osn_netif_apply() may be required before the change can take effect.

3.36.4.7 osn_netif_new()

```
osn_netif_t* osn_netif_new (
             const char * ifname )
```

Create a new instance of a network interface object.

Parameters

in	*ifname*	Interface name to which the netif instance will be bound to

Returns

This function returns NULL if an error occurs, otherwise a valid osn_netif_t object is returned.

Note

If the interface doesn't exist yet, this function may return success. The actual interface existence will be reported to the status callback.

3.36.4.8 osn_netif_state_set()

```
bool osn_netif_state_set (
             osn_netif_t * self,
             bool up )
```

Set the interface state. If up is set to true, the interface will be brought UP, otherwise it will be brought DOWN.

Parameters

in	*self*	A valid pointer to an osn_netif_t object
in	*up*	True if the interface state should be set to UP; for down use false

Returns

Returns true if the state option was successfully set.

Note

A call to osn_netif_apply() may be required before the change can take effect.

3.36.4.9 osn_netif_status_notify()

```
void osn_netif_status_notify (
            osn_netif_t * self,
            osn_netif_status_fn_t * fn )
```

Set the NETIF status callback.

Depending on the implementation, the status callback may be invoked periodically or whenever an interface status change has been detected. For maximum portability, the callback implementation should assume it can be called using either mode of operation.

Parameters

in	*self*	A valid pointer to an osn_netif_t object
in	*fn*	A pointer to the netif status callback handler

Note

The status change callback does not require a call to osn_netif_apply(). The callback handler must assume that it can be called any time in between osn_netif_status_notify() and osn_netif_del().

3.37 PPPoE

Classes

- struct osn_pppoe_status

Typedefs

- typedef struct osn_pppoe osn_pppoe_t
- typedef void osn_pppoe_status_fn_t(osn_pppoe_t *self, struct osn_pppoe_status *status)

Functions

- osn_pppoe_t * osn_pppoe_new (const char *ifname)
- bool osn_pppoe_del (osn_pppoe_t *self)
- bool osn_pppoe_parent_set (osn_pppoe_t *self, const char *parent_ifname)
- bool osn_pppoe_secret_set (osn_pppoe_t *self, const char *username, const char *password)
- bool osn_pppoe_apply (osn_pppoe_t *self)
- void osn_pppoe_data_set (osn_pppoe_t *self, void *data)
- void * osn_pppoe_data_get (osn_pppoe_t *self)
- void osn_pppoe_status_notify (osn_pppoe_t *self, osn_pppoe_status_fn_t *fn)

3.37.1 Detailed Description

OpenSync API for managing PPPoE links

3.37.2 Class Documentation

3.37.2.1 struct osn_pppoe_status

PPPoE link status reporting structure. This structure is used as a parameter to a osn_pppoe_status_fn_t type function. A status callback is typically registered with the osn_pppoe_status_notify() function.

Public Attributes

- const char * ps_ifname
- bool ps_exists
- bool ps_carrier
- osn_ip_addr_t ps_local_ip
- osn_ip_addr_t ps_remote_ip
- int ps_mtu

3.37.2.1.1 Member Data Documentation

3.37.2.1.1.1 ps_carrier

```
bool osn_pppoe_status::ps_carrier
```

The PPPoE interface is ready to send/receive packets

3.37.2.1.1.2 ps_exists

```
bool osn_pppoe_status::ps_exists
```

The PPPoE interface was created

3.37.2.1.1.3 ps_ifname

```
const char* osn_pppoe_status::ps_ifname
```

Interface name

3.37.2.1.1.4 ps_local_ip

```
osn_ip_addr_t osn_pppoe_status::ps_local_ip
```

Local IP address

3.37.2.1.1.5 ps_mtu

```
int osn_pppoe_status::ps_mtu
```

MTU of the interface

3.37.2.1.1.6 ps_remote_ip

```
osn_ip_addr_t osn_pppoe_status::ps_remote_ip
```

Remote IP address

3.37.3 Typedef Documentation

3.37.3.1 osn_pppoe_status_fn_t

```
typedef void osn_pppoe_status_fn_t(osn_pppoe_t *self, struct osn_pppoe_status *status)
```

Function callback type used for PPPoE status reporting. See osn_pppoe_status_notify for more details.

3.37.3.2 osn_pppoe_t

```
typedef struct osn_pppoe osn_pppoe_t
```

OSN PPPoE object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_pppoe_new() and must be destroyed using osn_pppoe_del().

3.37.4 Function Documentation

3.37.4.1 osn_pppoe_apply()

```
bool osn_pppoe_apply (
            osn_pppoe_t * self )
```

Apply configuration to the system.

This function applies the PPPoE data to the running system and creates the PPPoE interface.

Note

When this function returns, the running system may be still in an incomplete configuration state – this function just ensures that the configuration process has started.

3.37.4.2 osn_pppoe_data_get()

```
void* osn_pppoe_data_get (
            osn_pppoe_t * self )
```

Get the object `self` user data. If no user data was set, NULL will be returned.

Parameters

in	*self*	A valid pointer to an osn_pppoe_t object

Returns

Returns a pointer to user data previously set using osn_pppoe_data_set().

3.37.4.3 osn_pppoe_data_set()

```
void osn_pppoe_data_set (
            osn_pppoe_t * self,
            void * data )
```

Set the object `self` user data.

Parameters

in	*self*	A valid pointer to an osn_pppoe_t object
in	*data*	Pointer to user data

3.37.4.4 osn_pppoe_del()

```
bool osn_pppoe_del (
            osn_pppoe_t * self )
```

Destroy a valid osn_pppoe_t object.

Parameters

in	*self*	A valid pointer to an osn_pppoe_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

Note

All resources that were allocated during the lifetime of the object are freed.

3.37.4.5 osn_pppoe_new()

```
osn_pppoe_t* osn_pppoe_new (
             const char * ifname )
```

Create a new instance of a PPPoE interface object.

Parameters

in	*ifname*	Interface name of the PPPoE link

Returns

This function returns NULL if an error occurs, otherwise a valid osn_netif_t object is returned.

Note

The PPPoE interface may be created after osn_pppoe_apply() is called.

3.37.4.6 osn_pppoe_parent_set()

```
bool osn_pppoe_parent_set (
             osn_pppoe_t * self,
             const char * parent_ifname )
```

Set the parent interface; this interface will be used to create the PPPoE interface

Parameters

in	*self*	A valid pointer to an osn_pppoe_t object
in	*parent_ifname*	The parent interface name

This function must be called before osn_pppoe_apply(), otherwise the PPPoE interface creation will fail.

If this function is called multiple times, previous values are overwritten.

Returns

This function returns true on success. On error, false is returned.

3.37.4.7 osn_pppoe_secret_set()

```
bool osn_pppoe_secret_set (
            osn_pppoe_t * self,
            const char * username,
            const char * password )
```

Set credentials for this PPPoE connection.

Parameters

in	*self*	A valid pointer to an osn_pppoe_t object
in	*username*	Username to be used during PAP/CHAP authentication
in	*password*	Password to be used during PAP/CHAP authentication

If this function is called multiple times, previous credentials are overwritten.

Returns

This function returns true on success. On error, false is returned.

3.37.4.8 osn_pppoe_status_notify()

```
void osn_pppoe_status_notify (
            osn_pppoe_t * self,
            osn_pppoe_status_fn_t * fn )
```

Register a function callback that will be used for asynchronous PPPoE link status reporting. Depending on the implementation, the callback may be invoked before osn_pppoe_apply() is called (before the configuration is applied to the system).

3.38 VLAN

Typedefs

- typedef struct osn_vlan osn_vlan_t

Functions

- osn_vlan_t * osn_vlan_new (const char *ifname)
- bool osn_vlan_del (osn_vlan_t *self)
- bool osn_vlan_apply (osn_vlan_t *self)
- bool osn_vlan_parent_set (osn_vlan_t *self, const char *parent_ifname)
- bool osn_vlan_vid_set (osn_vlan_t *self, int vlanid)

3.38.1 Detailed Description

OpenSync API for managing VLAN interfaces

3.38.2 Typedef Documentation

3.38.2.1 osn_vlan_t

```
typedef struct osn_vlan osn_vlan_t
```

OSN VLAN object type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osn_vlan_new() and must be destroyed using osn_vlan_del().

3.38.3 Function Documentation

3.38.3.1 osn_vlan_apply()

```
bool osn_vlan_apply (
            osn_vlan_t * self )
```

Apply the interface VLAN configuration to the system. If not already created, this function will create the VLAN interface.

Note

When this function returns, the running system may be still in an incomplete configuration state – this function just ensures that the configuration process has started.

3.38.3.2 osn_vlan_del()

```
bool osn_vlan_del (
            osn_vlan_t * self )
```

Destroy a valid osn_vlan_t object.

Parameters

in	*self*	A valid pointer to an osn_vlan_t object

Returns

This function returns true on success. On error, false is returned. The input parameter should be considered invalid after this function returns, regardless of the error code.

Note

All resources that were allocated during the lifetime of the object are freed.

3.38.3.3 osn_vlan_new()

```
osn_vlan_t* osn_vlan_new (
            const char * ifname )
```

Create a new instance of a VLAN interface object.

Parameters

in	*ifname*	VLAN interface name

Returns

This function returns NULL if an error occurs, otherwise a valid osn_vlan_t object is returned.

Note

The VLAN interface may be created after osn_vlan_apply() is called.

3.38.3.4 osn_vlan_parent_set()

```
bool osn_vlan_parent_set (
            osn_vlan_t * self,
            const char * parent_ifname )
```

Set the parent interface; this interface will be used to create the VLAN interface

Parameters

in	*self*	A valid pointer to an osn_vlan_t object
in	*parent_ifname*	The parent interface name

This function must be called to set the parent interface name before osn_vlan_apply(), otherwise the VLAN interface creation will fail.

If this function is called multiple times, previous values are overwritten.

Returns

This function returns true on success. On error, false is returned.

3.38.3.5 osn_vlan_vid_set()

```
bool osn_vlan_vid_set (
            osn_vlan_t * self,
            int vlanid )
```

Set the VLAN ID of the interface.

Parameters

in	*self*	A valid pointer to an osn_vlan_t object
in	*vlanid*	The VLAN ID of the interface

A valid VLAN ID must be set before osn_vlan_apply() can succeed.

Returns

This function return true on success or false otherwise. On error, the previous value of the VLAN ID will be preserved.

3.39 OpenSync Platform API

Modules

- Unit API
- Thermal Management API
- Reboot API
- LED API
- Button API
- Upgrade API
- Persistent Storage API
- Download API
- Object Management API

3.39.1 Detailed Description

OpenSync Platform API and types

3.40 Unit API

Functions

- bool osp_unit_id_get (char *buff, size_t buffsz)
  Return device identification.
- bool osp_unit_serial_get (char *buff, size_t buffsz)
  Return device serial number.
- bool osp_unit_model_get (char *buff, size_t buffsz)
  Return device model.
- bool osp_unit_sku_get (char *buff, size_t buffsz)
  Return device stock keeping unit number.
- bool osp_unit_hw_revision_get (char *buff, size_t buffsz)
  Return hardware version number.
- bool osp_unit_platform_version_get (char *buff, size_t buffsz)
  Return platform version number.
- bool osp_unit_sw_version_get (char *buff, size_t buffsz)
  Return software version number.
- bool osp_unit_vendor_name_get (char *buff, size_t buffsz)
  Return vendor name.
- bool osp_unit_vendor_part_get (char *buff, size_t buffsz)
  Return vendor part number.
- bool osp_unit_manufacturer_get (char *buff, size_t buffsz)
  Return manufacturer name.
- bool osp_unit_factory_get (char *buff, size_t buffsz)
  Return factory name.
- bool osp_unit_mfg_date_get (char *buff, size_t buffsz)
  Return manufacturing date.

3.40.1 Detailed Description

OpenSync Unit API

3.40.2 Function Documentation

3.40.2.1 osp_unit_factory_get()

```
bool osp_unit_factory_get (
            char * buff,
            size_t buffsz )
```

Return factory name.

This function provides a null terminated byte string containing the factory name where the device was built. The factory name is part of the AWLAN_Node table. It is safe to return false here if not needed or unknown.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.2 osp_unit_hw_revision_get()

```
bool osp_unit_hw_revision_get (
             char * buff,
             size_t buffsz )
```

Return hardware version number.

This function provides a null terminated byte string containing the hardware version number. The hardware version is part of the AWLAN_Node table. If not needed this function should return false.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.3 osp_unit_id_get()

```
bool osp_unit_id_get (
             char * buff,
             size_t buffsz )
```

Return device identification.

This function provides a null terminated byte string containing the device identification. The device identification is part of the AWLAN_Node table. In the simplest implementation, this function may be the same as osp_unit_serial_get().

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.4 osp_unit_manufacturer_get()

```
bool osp_unit_manufacturer_get (
            char * buff,
            size_t buffsz )
```

Return manufacturer name.

This function provides a null terminated byte string containing the manufacturer name who built the device. The manufacturer name is part of the AWLAN_Node table. It is safe to return false here if not needed or unknown.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.5 osp_unit_mfg_date_get()

```
bool osp_unit_mfg_date_get (
            char * buff,
            size_t buffsz )
```

Return manufacturing date.

This function provides a null terminated byte string containing the date when the device was built. The date should be in a format "YYYY/WW", where YYYY stands for year, and WW stands for work week of the year. The manufacturing date is part of the AWLAN_Node table. It is safe to return false here if not needed or unknown.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.6 osp_unit_model_get()

```
bool osp_unit_model_get (
            char * buff,
            size_t buffsz )
```

Return device model.

This function provides a null terminated byte string containing the device model. The device model is a part of the AWLAN_Node table.

In the simplest implementation, this function may return the value of CONFIG_TARGET_MODEL.

It is safe to return false here. The TARGET_NAME will be used as a model name in that case.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.7 osp_unit_platform_version_get()

```
bool osp_unit_platform_version_get (
            char * buff,
            size_t buffsz )
```

Return platform version number.

This function provides a null terminated byte string containing the platform version number. The platform version number is part of the AWLAN_Node table. If not needed this function should return false.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.8 osp_unit_serial_get()

```
bool osp_unit_serial_get (
            char * buff,
            size_t buffsz )
```

Return device serial number.

This function provides a null terminated byte string containing the serial number. The serial number is part of the AW↩ LAN_Node table. For example, the serial number may be derived from the MAC address. Please see implementation inside osp_unit.c file for the reference.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.9 osp_unit_sku_get()

```
bool osp_unit_sku_get (
            char * buff,
            size_t buffsz )
```

Return device stock keeping unit number.

This function provides a null terminated byte string containing the stock keeping unit number. It is usually used by stores to track inventory. The SKU is part of the AWLAN_Node table.

If cloud doesn't support SKU for this target, this function should return false.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.10 osp_unit_sw_version_get()

```
bool osp_unit_sw_version_get (
            char * buff,
            size_t buffsz )
```

Return software version number.

This function provides a null terminated byte string containing the software version number. Expected format: VERS↩ ION-BUILD_NUMBER-gGITSHA-PROFILE Sample: 1.0.0.0-200-g1a2b3c-devel

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.11 osp_unit_vendor_name_get()

```
bool osp_unit_vendor_name_get (
            char * buff,
            size_t buffsz )
```

Return vendor name.

This function provides a null terminated byte string containing the device's vendor name. The vendor name is part of the AWLAN_Node table. It is safe to return false here if not needed.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.40.2.12 osp_unit_vendor_part_get()

```
bool osp_unit_vendor_part_get (
            char * buff,
            size_t buffsz )
```

Return vendor part number.

This function provides a null terminated byte string containing the device's vendor part number. The vendor part number is part of the AWLAN_Node table. It is safe to return false here if not needed.

Parameters

buff	pointer to a string buffer
buffsz	size of string buffer

Returns

true on success

3.41 Thermal Management API

Classes

- struct osp_tm_therm_state

Macros

- #define OSP_TM_TEMP_SRC_MAX (3)
- #define OSP_TM_TEMP_AVG_CNT (3)

Functions

- int osp_tm_init (const struct osp_tm_therm_state **tbl, unsigned int *therm_state_cnt, unsigned int *temp_↩ src_cnt, void **priv)
- void osp_tm_deinit (void *priv)
- bool osp_tm_is_temp_src_enabled (void *priv, int idx)
- const char * osp_tm_get_temp_src_name (void *priv, int idx)
- int osp_tm_get_temperature (void *priv, int idx, int *temp)
- int osp_tm_get_fan_rpm (void *priv, unsigned int *rpm)
- int osp_tm_set_fan_rpm (void *priv, unsigned int rpm)

3.41.1 Detailed Description

OpenSync Thermal Management API

3.41.2 Class Documentation

3.41.2.1 struct osp_tm_therm_state

Thermal state table element

Public Attributes

- int **temp_thrld** [OSP_TM_TEMP_SRC_MAX]
- unsigned int **radio_txchainmask** [OSP_TM_TEMP_SRC_MAX]
- unsigned int **fan_rpm**

3.41.3 Macro Definition Documentation

3.41.3.1 OSP_TM_TEMP_AVG_CNT

```
#define OSP_TM_TEMP_AVG_CNT (3)
```

Averaging window size

Measure running average of temperature over this number of temperature samples. This is a compromise between a low number of samples to react to fast rising temperature, and a high number of samples to react to bad temperature readings.

3.41.3.2 OSP_TM_TEMP_SRC_MAX

```
#define OSP_TM_TEMP_SRC_MAX (3)
```

Maximum number of temperature sources

3.41.4 Function Documentation

3.41.4.1 osp_tm_deinit()

```
void osp_tm_deinit (
            void * priv )
```

Thermal management subsystem cleanup

3.41.4.2 osp_tm_get_fan_rpm()

```
int osp_tm_get_fan_rpm (
            void * priv,
            unsigned int * rpm )
```

Return the current fan RPM

3.41.4.3 osp_tm_get_temp_src_name()

```
const char* osp_tm_get_temp_src_name (
            void * priv,
            int idx )
```

Return the name of the temperature source with index idx

3.41.4.4 osp_tm_get_temperature()

```
int osp_tm_get_temperature (
            void * priv,
            int idx,
            int * temp )
```

Return the temperature of the requested temperature source

3.41.4.5 osp_tm_init()

```
int osp_tm_init (
            const struct osp_tm_therm_state ** tbl,
            unsigned int * therm_state_cnt,
            unsigned int * temp_src_cnt,
            void ** priv )
```

Initialize thermal management subsystem

Should return a thermal states table, together with a count of thermal states and count of temperature sources. Thermal states table should go from the lowest thermal state to the highest. First element of the array should have lowest temperature thresholds. Last element of the array should have the highest temperature thresholds. If temperature rises above the highest temperature threshold, the device will be rebooted.

3.41.4.6 osp_tm_is_temp_src_enabled()

```
bool osp_tm_is_temp_src_enabled (
            void * priv,
            int idx )
```

Return true if temperature source with index idx is currently enabled

3.41.4.7 osp_tm_set_fan_rpm()

```
int osp_tm_set_fan_rpm (
            void * priv,
            unsigned int rpm )
```

Set the desired fan RPM

3.42 Reboot API

Enumerations

- enum osp_reboot_type {
  OSP_REBOOT_UNKNOWN,
  OSP_REBOOT_COLD_BOOT,
  OSP_REBOOT_POWER_CYCLE,
  OSP_REBOOT_WATCHDOG,
  OSP_REBOOT_CRASH,
  OSP_REBOOT_USER,
  OSP_REBOOT_DEVICE,
  OSP_REBOOT_HEALTH_CHECK,
  OSP_REBOOT_UPGRADE,
  OSP_REBOOT_THERMAL,
  OSP_REBOOT_CLOUD,
  OSP_REBOOT_CANCEL }

Functions

- bool osp_unit_reboot_ex (enum osp_reboot_type type, const char *reason, int ms_delay)
- bool osp_unit_factory_reboot (const char *reason, int ms_delay)
- bool osp_unit_reboot_get (enum osp_reboot_type *type, char *reason, ssize_t reason_sz)

3.42.1 Detailed Description

OpenSync Reboot API

3.42.2 Enumeration Type Documentation

3.42.2.1 osp_reboot_type

`enum osp_reboot_type`

Reboot type

Enumerator

OSP_REBOOT_UNKNOWN	Unknown reboot reason
OSP_REBOOT_COLD_BOOT	Power-on / cold boot
OSP_REBOOT_POWER_CYCLE	Power cycle or spontaneous reset
OSP_REBOOT_WATCHDOG	Watchdog triggered reboot
OSP_REBOOT_CRASH	Reboot due to kernel/system/driver crash
OSP_REBOOT_USER	Human triggered reboot (via shell or otherwise)
OSP_REBOOT_DEVICE	Device initiated reboot (upgrade, health check or otherwise)
OSP_REBOOT_HEALTH_CHECK	Health check failed
OSP_REBOOT_UPGRADE	Reboot due to an upgrade
OSP_REBOOT_THERMAL	Reboot due to a thermal event

3.42.3 Function Documentation

3.42.3.1 osp_unit_factory_reboot()

```
bool osp_unit_factory_reboot (
            const char * reason,
            int ms_delay )
```

Unit reboot & factory reset

Parameters

in	*reason*	Reboot reason (description)
in	*ms_delay*	Delay actual factory reboot in ms

Returns

true on success

3.42.3.2 osp_unit_reboot_ex()

```
bool osp_unit_reboot_ex (
            enum osp_reboot_type type,
            const char * reason,
            int ms_delay )
```

Unit reboot

Parameters

in	*type*	Reboot type (request source)
in	*reason*	Reboot reason (description)
in	*ms_delay*	Delay actual reboot in ms

Returns

true on success

Note

If the reboot type is OSP_REBOOT_CANCEL, the last record reboot record is invalidated.

3.42.3.3 osp_unit_reboot_get()

```
bool osp_unit_reboot_get (
             enum osp_reboot_type * type,
             char * reason,
             ssize_t reason_sz )
```

This function returns the last reboot type and reason

Parameters

out	*type*	Returns an enum of type osp_reboot_type
out	*reason*	Returns the reboot reason (as string)
out	*reason_sz*	Maximum size of `reason`

Returns

This function returns true if it was able to successfully detect the reboot type, or false in case of an error.

3.43 LED API

Macros

- #define OSP_LED_PRIORITY_DISABLE ((uint32_t)-1)
- #define OSP_LED_PRIORITY_DEFAULT ((uint32_t)-2)

Enumerations

- enum osp_led_state {
  OSP_LED_ST_IDLE = 0,
  OSP_LED_ST_ERROR,
  OSP_LED_ST_CONNECTED,
  OSP_LED_ST_CONNECTING,
  OSP_LED_ST_CONNECTFAIL,
  OSP_LED_ST_WPS,
  OSP_LED_ST_OPTIMIZE,
  OSP_LED_ST_LOCATE,
  OSP_LED_ST_HWERROR,
  OSP_LED_ST_THERMAL,
  OSP_LED_ST_BTCONNECTING,
  OSP_LED_ST_BTCONNECTED,
  OSP_LED_ST_BTCONNECTFAIL,
  OSP_LED_ST_UPGRADING,
  OSP_LED_ST_UPGRADED,
  OSP_LED_ST_UPGRADEFAIL,
  OSP_LED_ST_HWTEST,
  OSP_LED_ST_LAST }

Functions

- int osp_led_init (int *led_cnt)
- int osp_led_set_state (enum osp_led_state state, uint32_t priority)
- int osp_led_clear_state (enum osp_led_state state)
- int osp_led_reset (void)
- int osp_led_get_state (enum osp_led_state *state, uint32_t *priority)
- const char * osp_led_state_to_str (enum osp_led_state state)
- enum osp_led_state osp_led_str_to_state (const char *str)

3.43.1 Detailed Description

OpenSync LED API

3.43.2 Macro Definition Documentation

3.43.2.1 OSP_LED_PRIORITY_DEFAULT

```
#define OSP_LED_PRIORITY_DEFAULT ((uint32_t)-2)
```

Lowest priority

3.43.2.2 OSP_LED_PRIORITY_DISABLE

```
#define OSP_LED_PRIORITY_DISABLE ((uint32_t)-1)
```

LED state disabled

3.43.3 Enumeration Type Documentation

3.43.3.1 osp_led_state

```
enum osp_led_state
```

Available LED states

These are business logic level LED states, implemented by target layer.

Enumerator

OSP_LED_ST_IDLE	Idle (normal operation)
OSP_LED_ST_ERROR	Error state (generic)
OSP_LED_ST_CONNECTED	Connected
OSP_LED_ST_CONNECTING	Connecting
OSP_LED_ST_CONNECTFAIL	Failed to connect
OSP_LED_ST_WPS	WPS active
OSP_LED_ST_OPTIMIZE	Optimization in progress
OSP_LED_ST_LOCATE	Locating
OSP_LED_ST_HWERROR	Hardware fault
OSP_LED_ST_THERMAL	Thermal panic
OSP_LED_ST_BTCONNECTING	Bluetooth connecting
OSP_LED_ST_BTCONNECTED	Bluetooth connected
OSP_LED_ST_BTCONNECTFAIL	Bluetooth connection failed
OSP_LED_ST_UPGRADING	Upgrade in progress
OSP_LED_ST_UPGRADED	Upgrade finished
OSP_LED_ST_UPGRADEFAIL	Upgrade failed
OSP_LED_ST_HWTEST	Hardware test - FQC
OSP_LED_ST_LAST	(table sentinel)

3.43.4 Function Documentation

3.43.4.1 osp_led_clear_state()

```
int osp_led_clear_state (
            enum osp_led_state state )
```

Clear a LED state

If the specified state has the highest priority when being cleared, the next highest priority state is applied. If there are no states on the LED state stack, OSP_LED_ST_IDLE is applied.

Parameters

in	*state*	A previously set LED state

Returns

0 on success, -1 on error

3.43.4.2 osp_led_get_state()

```
int osp_led_get_state (
            enum osp_led_state * state,
            uint32_t * priority )
```

Get currently active LED state

Parameters

out	*state*	Current LED state
out	*priority*	Priority of the current state

Returns

0 on success, -1 on error

3.43.4.3 osp_led_init()

```
int osp_led_init (
            int * led_cnt )
```

Initialize the LED subsystem

Parameters

out	*led_cnt*	Number of LED's supported by the system

Returns

0 on success, -1 on error

3.43.4.4 osp_led_reset()

```
int osp_led_reset (
            void  )
```

Clear all LED states and set LED to 'Idle' state (OSP_LED_ST_IDLE)

Returns

0 on success, -1 on error

3.43.4.5 osp_led_set_state()

```
int osp_led_set_state (
            enum osp_led_state state,
            uint32_t priority )
```

Set LED to specified business level state (high-level LED API)

Parameters

in	*state*	LED state
in	*priority*	LED state priority – 0 is highest. A higher priority state overrides current LED behavior.

Returns

0 on success, -1 on error

3.43.4.6 osp_led_state_to_str()

```
const char* osp_led_state_to_str (
            enum osp_led_state state )
```

Get the textual representation of the given state

Parameters

in	*state*	LED state to convert to string

Returns

null-terminated string

3.43.4.7 osp_led_str_to_state()

```
enum osp_led_state osp_led_str_to_state (
            const char * str )
```

Convert string to LED state

Parameters

in	*str*	null-terminated string to convert

Returns

state or OSP_LED_ST_LAST on failure

3.44 Button API

Classes

- struct osp_btn_event

Typedefs

- typedef void(∗ osp_btn_cb) (void ∗obj, enum osp_btn_name name, const struct osp_btn_event ∗event)

Enumerations

- enum osp_btn_name {
  OSP_BTN_NAME_RESET = (1 << 0),
  OSP_BTN_NAME_WPS = (1 << 1) }

Functions

- bool osp_btn_get_caps (uint32_t ∗caps)
- bool osp_btn_register (osp_btn_cb cb, void ∗obj)

3.44.1 Detailed Description

OpenSync Button API

3.44.2 Class Documentation

3.44.2.1 struct osp_btn_event

Definition of an event associated with a button

Example 1: Button is pushed

- pushed = true
- duration = 0
- double_click = false

Example 2: Button is double click

- pushed = false
- duration = 0

- double_click = true

Example 3: Button is released after 1 second

- pushed = false
- duration = 1000
- double_click = false

Example 4: Button is released after 5 seconds

- pushed = false
- duration = 5000
- double_click = false

Public Attributes

- bool pushed
- unsigned int duration
- bool double_click

3.44.2.1.1 Member Data Documentation

3.44.2.1.1.1 double_click

```
bool osp_btn_event::double_click
```

True if the button is pushed and released two times in less than 1000 milliseconds

Valid only when the button is released.

3.44.2.1.1.2 duration

```
unsigned int osp_btn_event::duration
```

Duration in milliseconds of pressing the button

Valid only when the button is released and it was not a double click.

3.44.2.1.1.3 pushed

```
bool osp_btn_event::pushed
```

True if the button is pushed, false if the button is released

3.44.3 Typedef Documentation

3.44.3.1 osp_btn_cb

```
typedef void(* osp_btn_cb) (void *obj, enum osp_btn_name name, const struct osp_btn_event *event)
```

Callback called by the target layer when an event is received on a button

Parameters

in	*obj*	Pointer to the object that was supplied when the callback was registered (osp_btn_register call)
in	*name*	Button associated with the event
in	*event*	Details of the button event

3.44.4 Enumeration Type Documentation

3.44.4.1 osp_btn_name

```
enum osp_btn_name
```

Enumeration of buttons supported by OpenSync

Enumerator

OSP_BTN_NAME_RESET	Factory reset button
OSP_BTN_NAME_WPS	WiFi WPS button

3.44.5 Function Documentation

3.44.5.1 osp_btn_get_caps()

```
bool osp_btn_get_caps (
            uint32_t * caps )
```

Get the capabilities related to the buttons

Parameters

out	*caps*	Bitmask of buttons supported by the target You can test if a button is supported by testing the bitmask For example, to test if the reset button is supported by the target, you can test (caps & OSP_BTN_NAME_RESET)

Returns

true on success

3.44.5.2 osp_btn_register()

```
bool osp_btn_register (
            osp_btn_cb cb,
            void * obj )
```

Register the callback to receive button events

Parameters

in	*cb*	Callback called by the target layer when an event is received on a button. If callback is NULL, the target must unregister the previous one for this specific obj.
in	*obj*	User pointer which will be given back when the callback will be called

Returns

true on success

3.45 Upgrade API

Typedefs

- typedef void(∗ osp_upg_cb) (const osp_upg_op_t op, const osp_upg_status_t status, uint8_t completed)

Enumerations

- enum osp_upg_op_t {
  OSP_UPG_DL,
  OSP_UPG_UPG }
- enum osp_upg_status_t {
  OSP_UPG_OK = 0,
  OSP_UPG_ARGS = 1,
  OSP_UPG_URL = 3,
  OSP_UPG_DL_FW = 4,
  OSP_UPG_DL_MD5 = 5,
  OSP_UPG_MD5_FAIL = 6,
  OSP_UPG_IMG_FAIL = 7,
  OSP_UPG_FL_ERASE = 8,
  OSP_UPG_FL_WRITE = 9,
  OSP_UPG_FL_CHECK = 10,
  OSP_UPG_BC_SET = 11,
  OSP_UPG_APPLY = 12,
  OSP_UPG_BC_ERASE = 14,
  OSP_UPG_SU_RUN = 15,
  OSP_UPG_DL_NOFREE = 16,
  OSP_UPG_WRONG_PARAM = 17,
  OSP_UPG_INTERNAL = 18 }

Functions

- bool osp_upg_check_system (void)
- bool osp_upg_dl (char ∗url, uint32_t timeout, osp_upg_cb dl_cb)
- bool osp_upg_upgrade (char ∗password, osp_upg_cb upg_cb)
- bool osp_upg_commit (void)
- int osp_upg_errno (void)

3.45.1 Detailed Description

OpenSync Upgrade API

3.45.2 Typedef Documentation

3.45.2.1 osp_upg_cb

```
typedef void(* osp_upg_cb) (const osp_upg_op_t op, const osp_upg_status_t status, uint8_t completed)
```

Callback invoked by target layer during download & upgrade process

Parameters

in	*op*	- operation: download, download CS file or upgrade
in	*status*	status
in	*completed*	percentage of completed work 0 - 100%

3.45.3 Enumeration Type Documentation

3.45.3.1 osp_upg_op_t

`enum osp_upg_op_t`

Type of upgrade operations

Enumerator

OSP_UPG_DL	Download of the upgrade file
OSP_UPG_UPG	Upgrade process

3.45.3.2 osp_upg_status_t

`enum osp_upg_status_t`

Upgrade operations status

Enumerator

OSP_UPG_OK	Success
OSP_UPG_ARGS	Wrong arguments (app error)
OSP_UPG_URL	Error setting url
OSP_UPG_DL_FW	DL of FW image failed
OSP_UPG_DL_MD5	DL of ∗.md5 sum failed
OSP_UPG_MD5_FAIL	md5 CS failed or platform
OSP_UPG_IMG_FAIL	Image check failed
OSP_UPG_FL_ERASE	Flash erase failed
OSP_UPG_FL_WRITE	Flash write failed
OSP_UPG_FL_CHECK	Flash verification failed
OSP_UPG_BC_SET	New FW commit failed
OSP_UPG_APPLY	Applying new FW failed

Enumerator

OSP_UPG_BC_ERASE	Clean FW commit info failed
OSP_UPG_SU_RUN	Upgrade in progress running
OSP_UPG_DL_NOFREE	Not enough free space on unit
OSP_UPG_WRONG_PARAM	Wrong flashing parameters
OSP_UPG_INTERNAL	Internal error

3.45.4 Function Documentation

3.45.4.1 osp_upg_check_system()

```
bool osp_upg_check_system (
            void  )
```

Check system requirements for upgrade, like no upgrade in progress, available flash space etc.

3.45.4.2 osp_upg_commit()

```
bool osp_upg_commit (
            void  )
```

On dual-boot system, flag the newly flashed image as the active one. This can be a no-op on single image systems.

3.45.4.3 osp_upg_dl()

```
bool osp_upg_dl (
            char * url,
            uint32_t timeout,
            osp_upg_cb dl_cb )
```

Download an image suitable for upgrade from `uri` and store it locally. Upon download and verification completion, invoke the `dl_cb` callback.

3.45.4.4 osp_upg_errno()

```
int osp_upg_errno (
            void  )
```

Return a more detailed error code related to a failed osp_upg_∗() function call. See osp_upg_status_t for a detailed list of error codes.

3.45.4.5 osp_upg_upgrade()

```
bool osp_upg_upgrade (
            char * password,
            osp_upg_cb upg_cb )
```

Write the previously downloaded image to the system. If the image is encrypted, a password must be specified in `password`.

After the image was successfully applied, the `upg_cb` callback is invoked.

3.46 Persistent Storage API

Macros

- #define OSP_PS_READ (1 << 0)
- #define OSP_PS_WRITE (1 << 1)
- #define OSP_PS_PRESERVE (1 << 2)
- #define OSP_PS_RDWR (OSP_PS_READ | OSP_PS_WRITE)

Typedefs

- typedef struct osp_ps osp_ps_t

Functions

- osp_ps_t * osp_ps_open (const char *store, int flags)
- bool osp_ps_close (osp_ps_t *ps)
- ssize_t osp_ps_set (osp_ps_t *ps, const char *key, void *value, size_t value_sz)
- ssize_t osp_ps_get (osp_ps_t *ps, const char *key, void *value, size_t value_sz)
- bool osp_ps_erase (osp_ps_t *ps)
- bool osp_ps_sync (osp_ps_t *ps)

3.46.1 Detailed Description

OpenSync Persistent Storage API

3.46.2 Macro Definition Documentation

3.46.2.1 OSP_PS_PRESERVE

```
#define OSP_PS_PRESERVE (1 << 2)
```

Preserve store across upgrades

3.46.2.2 OSP_PS_RDWR

```
#define OSP_PS_RDWR (OSP_PS_READ | OSP_PS_WRITE)
```

Read-write access

3.46.2.3 OSP_PS_READ

```
#define OSP_PS_READ (1 << 0)
```

Flags for Read mode

3.46.2.4 OSP_PS_WRITE

```
#define OSP_PS_WRITE (1 << 1)
```

Write mode

3.46.3 Typedef Documentation

3.46.3.1 osp_ps_t

```
typedef struct osp_ps osp_ps_t
```

OSP Persistent Storage handle type

This is an opaque type. The actual structure implementation is hidden and is platform dependent. A new instance of the object can be obtained by calling osp_ps_open() and must be destroyed using osp_ps_close().

3.46.4 Function Documentation

3.46.4.1 osp_ps_close()

```
bool osp_ps_close (
            osp_ps_t * ps )
```

Release the `ps` handle and clean up any resources associated with it. Pending data will be flushed to persistent storage.

Parameters

`in`	*ps*	Store – valid object returned by osp_ps_open()

Note

This function automatically syncs data to persistent storage as if osp_ps_sync() was called.

3.46.4.2 osp_ps_erase()

```
bool osp_ps_erase (
            osp_ps_t * ps )
```

Erase content of store `ps` (delete all keys and their values)

Parameters

in	*ps*	Store – valid object returned by osp_ps_open() with the flag OSP_PS_WRITE

Note

Stores opened with the same name but with or without the OPS_PS_PRESERVE flag are different stores.

Returns

This function returns true on success, or false if any errors were encountered. If false is returned, it should be assumed that store was not erased.

Note

This function does not guarantee that the data was deleted from persistent store. To ensure that the change hits the storage, a call to osp_ps_sync() or osp_ps_close() is required.

3.46.4.3 osp_ps_get()

```
ssize_t osp_ps_get (
            osp_ps_t * ps,
            const char * key,
            void * value,
            size_t value_sz )
```

Retrieve data associated with `key` from store

Parameters

in	*ps*	Store – valid object returned by osp_ps_open() with the flag OSP_PS_READ
in	*key*	Key to retrieve
out	*value*	Pointer to value data to store; can be NULL if `value_sz` is 0
in	*value_sz*	Maximum length of `value`, data will be truncated if the actual size exceeds `value_sz`

Returns

Return the data size associated with the key, a value of <0 on error or 0 if they key was not found.

Note

If `value_sz` is less than the actual key data, the data will be truncated. However, the return value will still be the actual data size.

3.46.4.4 osp_ps_open()

```
osp_ps_t* osp_ps_open (
            const char * store,
            int flags )
```

Open store `store`

Parameters

in	*store*	Store name
in	*flags*	Read/write mode – this may determine the type of lock used

Returns

Return a valid handle to a store

Note

To enable concurrent access from multiple processes, the store may be protected by means of global locks. This means that the time between a osp_ps_open() and osp_ps_close() must be kept at a minimum.

3.46.4.5 osp_ps_set()

```
ssize_t osp_ps_set (
            osp_ps_t * ps,
            const char * key,
            void * value,
            size_t value_sz )
```

Store value or delete value data associated with key `key`

Parameters

in	*ps*	Store – valid object returned by osp_ps_open() with the flag OSP_PS_WRITE
in	*key*	Key value
in	*value*	Pointer to value data to store; can be NULL if value_sz is 0
in	*value_sz*	Value data length, if 0 the key is deleted

Returns

This function returns the number of bytes stored, <0 on error or 0 if the entry was successfully deleted.

Note

This function does not guarantee that the data was saved to persistent store. To ensure that data hits the storage, a call to osp_ps_sync() or osp_ps_close() is required.

3.46.4.6 osp_ps_sync()

```
bool osp_ps_sync (
            osp_ps_t * ps )
```

Flush all dirty data to persistent storage. When this function returns, the data written by osp_ps_set() should be considered safely stored.

Parameters

in	*ps*	Store – valid object returned by osp_ps_open() with the flag OSP_PS_WRITE

Returns

This function returns true on success, or false if any errors were encountered. If false is returned, it should be assumed that data loss may occur.

3.47 Download API

Typedefs

- typedef void(∗ osp_dl_cb) (const enum osp_dl_status status, void ∗cb_ctx)

Enumerations

- enum osp_dl_status {
  OSP_DL_OK = 0,
  OSP_DL_DOWNLOAD_FAILED,
  OSP_DL_ERROR }

Functions

- bool osp_dl_download (char ∗url, char ∗dst_path, int timeout, osp_dl_cb dl_cb, void ∗cb_ctx)

3.47.1 Detailed Description

OpenSync Download API

3.47.2 Typedef Documentation

3.47.2.1 osp_dl_cb

```
typedef void(* osp_dl_cb) (const enum osp_dl_status status, void *cb_ctx)
```

Complete download callback function

Parameters

in	*status*	Status of finished download
in	*cb_ctx*	Context struct of osp_dl_download caller.

3.47.3 Enumeration Type Documentation

3.47.3.1 osp_dl_status

```
enum osp_dl_status
```

Enum osp_dl_status for status reporting

Enumerator

OSP_DL_OK	Download OK.
OSP_DL_DOWNLOAD_FAILED	Download failed.
OSP_DL_ERROR	General download error.

3.47.4 Function Documentation

3.47.4.1 osp_dl_download()

```
bool osp_dl_download (
            char * url,
            char * dst_path,
            int timeout,
            osp_dl_cb dl_cb,
            void * cb_ctx )
```

Function to download a file from `url` to `dst_path`. Non-blocking implementation is expected. After a successful download, a failure, or an expired timeout, it is expected that dl_cb callback is called with the status.

Parameters

in	*url*	URL of file to download
in	*dst_path*	Path where to download the file to
in	*timeout*	Timeout for the download operation
in	*dl_cb*	Callback for when downloading is finished, or failure or a timeout occurred
in	*cb_ctx*	Caller context struct that is passed in dl_cb callback

Returns

true if download is started successfully

3.48 Object Management API

Functions

- bool osp_objm_install (char ∗path, char ∗name, char ∗version)
- bool osp_objm_remove (char ∗name, char ∗version)
- bool osp_objm_path (char ∗buf, size_t buffsz, char ∗name, char ∗version)

3.48.1 Detailed Description

OpenSync Object Management API

3.48.2 Function Documentation

3.48.2.1 osp_objm_install()

```
bool osp_objm_install (
            char * path,
            char * name,
            char * version )
```

Install object to object storage

Parameters

in	*path*	Path where the file for installation is located
in	*name*	Name of the object
in	*version*	Version of object

Returns

true if install is successful

3.48.2.2 osp_objm_path()

```
bool osp_objm_path (
            char * buf,
            size_t buffsz,
```

```
            char * name,
            char * version )
```

Get path on filesystem where installed object is available

Parameters

out	*buf*	Buffer in which path of object is returned
in	*buffsz*	Size of the provided buffer
in	*name*	Name of the object
in	*version*	Version of object

Returns

true if buf is populated with path

3.48.2.3 osp_objm_remove()

```
bool osp_objm_remove (
            char * name,
            char * version )
```

Remove object from object storage

Parameters

in	*name*	Name of the object
in	*version*	Version of object

Returns

true if removal is successful

Chapter 4

File Documentation

4.1 osn_dhcp.h File Reference

OpenSync DHCPv4.

```
#include <stdbool.h>
#include "const.h"
#include "osn_types.h"
```

Classes

- struct osn_dhcp_server_cfg
- struct osn_dhcp_server_lease
- struct osn_dhcp_server_status

Macros

- #define OSN_DHCP_FINGERPRINT_MAX 256
- #define OSN_DHCP_VENDORCLASS_MAX 256
- #define OSN_DHCP_SERVER_CFG_INIT
- #define OSN_DHCP_SERVER_LEASE_INIT

Typedefs

- typedef struct osn_dhcp_client osn_dhcp_client_t
- typedef void osn_dhcp_client_error_fn_t(osn_dhcp_client_t *self)
- typedef bool osn_dhcp_client_opt_notify_fn_t(osn_dhcp_client_t *self, enum osn_notify hint, const char *key, const char *value)
- typedef struct osn_dhcp_server osn_dhcp_server_t
- typedef void osn_dhcp_server_status_fn_t(osn_dhcp_server_t *self, struct osn_dhcp_server_status *status)
- typedef void osn_dhcp_server_error_fn_t(osn_dhcp_server_t *self)

Enumerations

- enum osn_notify {
  NOTIFY_UPDATE,
  NOTIFY_DELETE,
  NOTIFY_SYNC,
  NOTIFY_FLUSH }
- enum osn_dhcp_option {
  DHCP_OPTION_SUBNET_MASK = 1,
  DHCP_OPTION_ROUTER = 3,
  DHCP_OPTION_DNS_SERVERS = 6,
  DHCP_OPTION_HOSTNAME = 12,
  DHCP_OPTION_DOMAIN_NAME = 15,
  DHCP_OPTION_BCAST_ADDR = 28,
  DHCP_OPTION_VENDOR_SPECIFIC = 43,
  DHCP_OPTION_ADDRESS_REQUEST = 50,
  DHCP_OPTION_LEASE_TIME = 51,
  DHCP_OPTION_MSG_TYPE = 53,
  DHCP_OPTION_PARAM_LIST = 55,
  DHCP_OPTION_VENDOR_CLASS = 60,
  DHCP_OPTION_DOMAIN_SEARCH = 119,
  DHCP_OPTION_OSYNC_SWVER = 225,
  DHCP_OPTION_OSYNC_PROFILE = 226,
  DHCP_OPTION_OSYNC_SERIAL_OPT = 227,
  DHCP_OPTION_MAX = 256 }

Functions

- osn_dhcp_client_t * osn_dhcp_client_new (const char *ifname)
- bool osn_dhcp_client_del (osn_dhcp_client_t *self)
- bool osn_dhcp_client_start (osn_dhcp_client_t *self)
- bool osn_dhcp_client_stop (osn_dhcp_client_t *self)
- bool osn_dhcp_client_opt_request (osn_dhcp_client_t *self, enum osn_dhcp_option opt, bool request)
- bool osn_dhcp_client_opt_set (osn_dhcp_client_t *self, enum osn_dhcp_option opt, const char *value)
- bool osn_dhcp_client_opt_get (osn_dhcp_client_t *self, enum osn_dhcp_option opt, bool *request, const char **value)
- bool osn_dhcp_client_opt_notify_set (osn_dhcp_client_t *self, osn_dhcp_client_opt_notify_fn_t *fn)
- bool osn_dhcp_client_error_fn_set (osn_dhcp_client_t *self, osn_dhcp_client_error_fn_t *fn)
- bool osn_dhcp_client_vendorclass_set (osn_dhcp_client_t *self, const char *vendorspec)
- bool osn_dhcp_client_state_get (osn_dhcp_client_t *self, bool *enabled)
- void osn_dhcp_client_data_set (osn_dhcp_client_t *self, void *data)
- void * osn_dhcp_client_data_get (osn_dhcp_client_t *self)
- osn_dhcp_server_t * osn_dhcp_server_new (const char *ifname)
- bool osn_dhcp_server_del (osn_dhcp_server_t *self)
- void osn_dhcp_server_data_set (osn_dhcp_server_t *self, void *data)
- void * osn_dhcp_server_data_get (osn_dhcp_server_t *self)
- bool osn_dhcp_server_cfg_set (osn_dhcp_server_t *self, struct osn_dhcp_server_cfg *cfg)
- bool osn_dhcp_server_range_add (osn_dhcp_server_t *self, osn_ip_addr_t start, osn_ip_addr_t stop)
- bool osn_dhcp_server_range_del (osn_dhcp_server_t *self, osn_ip_addr_t start, osn_ip_addr_t stop)
- bool osn_dhcp_server_option_set (osn_dhcp_server_t *self, enum osn_dhcp_option opt, const char *value)
- void osn_dhcp_server_error_notify (osn_dhcp_server_t *self, osn_dhcp_server_error_fn_t *fn)
- void osn_dhcp_server_status_notify (osn_dhcp_server_t *self, osn_dhcp_server_status_fn_t *fn)
- bool osn_dhcp_server_reservation_add (osn_dhcp_server_t *self, osn_mac_addr_t macaddr, osn_ip_addr_t ip4addr, const char *hostname)
- bool osn_dhcp_server_reservation_del (osn_dhcp_server_t *self, osn_mac_addr_t macaddr)
- bool osn_dhcp_server_apply (osn_dhcp_server_t *self)

4.1.1 Detailed Description

OpenSync DHCPv4.

4.2 osn_dhcpv6.h File Reference

OpenSync DHCPv6.

```
#include "osn_inet6.h"
```

Classes

- struct osn_dhcpv6_client_status
- struct osn_dhcpv6_server_prefix
- struct osn_dhcpv6_server_lease
- struct osn_dhcpv6_server_status

Macros

- #define OSN_DHCP_HOSTNAME_LEN 64
- #define OSN_DHCP_OPTIONS_MAX 256

Typedefs

- typedef struct osn_dhcpv6_client osn_dhcpv6_client_t
- typedef void osn_dhcpv6_client_status_fn_t(osn_dhcpv6_client_t *self, struct osn_dhcpv6_client_status *status)
- typedef struct osn_dhcpv6_server osn_dhcpv6_server_t
- typedef void osn_dhcpv6_server_status_fn_t(osn_dhcpv6_server_t *d6s, struct osn_dhcpv6_server_status *status)

Functions

- osn_dhcpv6_client_t * osn_dhcpv6_client_new (const char *ifname)
- bool osn_dhcpv6_client_del (osn_dhcpv6_client_t *self)
- bool osn_dhcpv6_client_set (osn_dhcpv6_client_t *self, bool request_address, bool request_prefixes, bool rapid_commit, bool renew)
- bool osn_dhcpv6_client_option_request (osn_dhcpv6_client_t *self, int tag)
- bool osn_dhcpv6_client_option_send (osn_dhcpv6_client_t *self, int tag, const char *value)
- void osn_dhcpv6_client_status_notify (osn_dhcpv6_client_t *self, osn_dhcpv6_client_status_fn_t *fn)
- bool osn_dhcpv6_client_apply (osn_dhcpv6_client_t *self)
- void osn_dhcpv6_client_data_set (osn_dhcpv6_client_t *self, void *data)
- void * osn_dhcpv6_client_data_get (osn_dhcpv6_client_t *self)
- osn_dhcpv6_server_t * osn_dhcpv6_server_new (const char *iface)
- bool osn_dhcpv6_server_del (osn_dhcpv6_server_t *self)
- bool osn_dhcpv6_server_apply (osn_dhcpv6_server_t *self)
- void osn_dhcpv6_server_data_set (osn_dhcpv6_server_t *self, void *data)
- void * osn_dhcpv6_server_data_get (osn_dhcpv6_server_t *self)
- bool osn_dhcpv6_server_prefix_add (osn_dhcpv6_server_t *self, struct osn_dhcpv6_server_prefix *prefix)
- bool osn_dhcpv6_server_prefix_del (osn_dhcpv6_server_t *self, struct osn_dhcpv6_server_prefix *prefix)
- bool osn_dhcpv6_server_option_send (osn_dhcpv6_server_t *self, int tag, const char *value)
- bool osn_dhcpv6_server_lease_add (osn_dhcpv6_server_t *self, struct osn_dhcpv6_server_lease *lease)
- bool osn_dhcpv6_server_lease_del (osn_dhcpv6_server_t *self, struct osn_dhcpv6_server_lease *lease)
- bool osn_dhcpv6_server_status_notify (osn_dhcpv6_server_t *self, osn_dhcpv6_server_status_fn_t *fn)

4.2.1 Detailed Description

OpenSync DHCPv6.

4.3 osn_inet.h File Reference

OpenSync IPv4.

```
#include <stdbool.h>
#include "osn_types.h"
```

Classes

- struct osn_ip_status
- struct osn_route_status

Macros

- #define OSN_ROUTE_STATUS_INIT

Typedefs

- typedef struct osn_ip osn_ip_t
- typedef void osn_ip_status_fn_t(osn_ip_t *ip, struct osn_ip_status *status)
- typedef struct osn_route osn_route_t
- typedef bool osn_route_status_fn_t(osn_route_t *self, struct osn_route_status *rts, bool remove)

Functions

- osn_ip_t * osn_ip_new (const char *ifname)
- bool osn_ip_del (osn_ip_t *ip)
- bool osn_ip_addr_add (osn_ip_t *ip, const osn_ip_addr_t *addr)
- bool osn_ip_addr_del (osn_ip_t *ip, const osn_ip_addr_t *addr)
- bool osn_ip_dns_add (osn_ip_t *ip, const osn_ip_addr_t *dns)
- bool osn_ip_dns_del (osn_ip_t *ip, const osn_ip_addr_t *dns)
- bool osn_ip_route_gw_add (osn_ip_t *ip, const osn_ip_addr_t *src, const osn_ip_addr_t *gw)
- bool osn_ip_route_gw_del (osn_ip_t *ip, const osn_ip_addr_t *src, const osn_ip_addr_t *gw)
- void osn_ip_data_set (osn_ip_t *ip, void *data)
- void * osn_ip_data_get (osn_ip_t *ip)
- void osn_ip_status_notify (osn_ip_t *ip, osn_ip_status_fn_t *fn)
- bool osn_ip_apply (osn_ip_t *ip)
- osn_route_t * osn_route_new (const char *ifname)
- bool osn_route_del (osn_route_t *self)
- bool osn_route_status_notify (osn_route_t *self, osn_route_status_fn_t *fn)
- void osn_route_data_set (osn_route_t *self, void *data)
- void * osn_route_data_get (osn_route_t *self)

4.3.1 Detailed Description

OpenSync IPv4.

4.4 osn_inet6.h File Reference

OpenSync IPv6.

```
#include <netinet/in.h>
#include <limits.h>
#include "osn_types.h"
```

Classes

- struct osn_ip6_neigh
- struct osn_ip6_status
- struct osn_ip6_radv_options

Macros

- #define OSN_IP6_RADV_OPTIONS_INIT

Typedefs

- typedef struct osn_ip6 osn_ip6_t
- typedef void osn_ip6_status_fn_t(osn_ip6_t *self, struct osn_ip6_status *status)
- typedef struct osn_ip6_radv osn_ip6_radv_t

Functions

- osn_ip6_t * osn_ip6_new (const char *ifname)
- bool osn_ip6_del (osn_ip6_t *self)
- bool osn_ip6_apply (osn_ip6_t *self)
- bool osn_ip6_addr_add (osn_ip6_t *self, const osn_ip6_addr_t *addr)
- bool osn_ip6_addr_del (osn_ip6_t *self, const osn_ip6_addr_t *addr)
- bool osn_ip6_dns_add (osn_ip6_t *self, const osn_ip6_addr_t *dns)
- bool osn_ip6_dns_del (osn_ip6_t *self, const osn_ip6_addr_t *dns)
- void osn_ip6_status_notify (osn_ip6_t *self, osn_ip6_status_fn_t *fn)
- void osn_ip6_data_set (osn_ip6_t *self, void *data)
- void * osn_ip6_data_get (osn_ip6_t *self)
- osn_ip6_radv_t * osn_ip6_radv_new (const char *ifname)
- bool osn_ip6_radv_del (osn_ip6_radv_t *self)
- bool osn_ip6_radv_set (osn_ip6_radv_t *self, const struct osn_ip6_radv_options *opts)
- bool osn_ip6_radv_add_prefix (osn_ip6_radv_t *self, const osn_ip6_addr_t *prefix, bool autonomous, bool on-link)
- bool osn_ip6_radv_del_prefix (osn_ip6_radv_t *self, const osn_ip6_addr_t *prefix)
- bool osn_ip6_radv_add_rdnss (osn_ip6_radv_t *self, const osn_ip6_addr_t *dns)
- bool osn_ip6_radv_del_rdnss (osn_ip6_radv_t *self, const osn_ip6_addr_t *dns)
- bool osn_ip6_radv_add_dnssl (osn_ip6_radv_t *self, char *sl)
- bool osn_ip6_radv_del_dnssl (osn_ip6_radv_t *self, char *sl)
- bool osn_ip6_radv_apply (osn_ip6_radv_t *self)

4.4.1 Detailed Description

OpenSync IPv6.

4.5 osn_netif.h File Reference

Network Interface L2 Abstraction.

```
#include <stdbool.h>
#include "osn_types.h"
```

Classes

- struct osn_netif_status

Typedefs

- typedef struct osn_netif osn_netif_t
- typedef void osn_netif_status_fn_t(osn_netif_t *self, struct osn_netif_status *status)

Functions

- osn_netif_t * osn_netif_new (const char *ifname)
- bool osn_netif_del (osn_netif_t *self)
- void osn_netif_data_set (osn_netif_t *self, void *data)
- void * osn_netif_data_get (osn_netif_t *self)
- void osn_netif_status_notify (osn_netif_t *self, osn_netif_status_fn_t *fn)
- bool osn_netif_apply (osn_netif_t *self)
- bool osn_netif_state_set (osn_netif_t *self, bool up)
- bool osn_netif_mtu_set (osn_netif_t *self, int mtu)
- bool osn_netif_hwaddr_set (osn_netif_t *self, osn_mac_addr_t hwaddr)

4.5.1 Detailed Description

Network Interface L2 Abstraction.

This API provides management of L2 Ethernet-like interfaces.

4.6 osn_pppoe.h File Reference

OpenSync PPPoE Interface Abstraction.

```
#include <stdbool.h>
#include "osn_types.h"
```

Classes

- struct osn_pppoe_status

Typedefs

- typedef struct osn_pppoe osn_pppoe_t
- typedef void osn_pppoe_status_fn_t(osn_pppoe_t *self, struct osn_pppoe_status *status)

Functions

- osn_pppoe_t * osn_pppoe_new (const char *ifname)
- bool osn_pppoe_del (osn_pppoe_t *self)
- bool osn_pppoe_parent_set (osn_pppoe_t *self, const char *parent_ifname)
- bool osn_pppoe_secret_set (osn_pppoe_t *self, const char *username, const char *password)
- bool osn_pppoe_apply (osn_pppoe_t *self)
- void osn_pppoe_data_set (osn_pppoe_t *self, void *data)
- void * osn_pppoe_data_get (osn_pppoe_t *self)
- void osn_pppoe_status_notify (osn_pppoe_t *self, osn_pppoe_status_fn_t *fn)

4.6.1 Detailed Description

OpenSync PPPoE Interface Abstraction.

4.7 osn_types.h File Reference

OpenSync Networking Common Types.

```
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdbool.h>
#include <stdint.h>
#include <stddef.h>
#include <string.h>
#include <stdio.h>
```

Classes

- struct osn_ip_addr
- struct osn_ip6_addr
- struct osn_mac_addr

Macros

- #define OSN_IP_ADDR_INIT
- #define OSN_IP_ADDR_LEN sizeof("255.255.255.255/32")
- #define PRI_osn_ip_addr "%s"
- #define FMT_osn_ip_addr(x) (__FMT_osn_ip_addr((char[OSN_IP_ADDR_LEN]){0}, OSN_IP_ADDR_LEN, &x))
- #define OSN_IP6_ADDR_INIT
- #define OSN_IP6_ADDR_LEN sizeof("1111:2222:3333:4444:5555:6666:7777:8888/128,2147483648,2147483648")
- #define PRI_osn_ip6_addr "%s"
- #define FMT_osn_ip6_addr(x) (__FMT_osn_ip6_addr((char[OSN_IP6_ADDR_LEN]){0}, OSN_IP6_ADDR_LEN, &x))
- #define OSN_MAC_ADDR_LEN sizeof("11:22:33:44:55:66")
- #define OSN_MAC_ADDR_INIT (osn_mac_addr_t){ .ma_addr = { 0 }, }
- #define PRI_osn_mac_addr "%02x:%02x:%02x:%02x:%02x:%02x"
- #define FMT_osn_mac_addr(x)

Typedefs

- typedef struct osn_ip_addr osn_ip_addr_t
- typedef struct osn_ip6_addr osn_ip6_addr_t
- typedef struct osn_mac_addr osn_mac_addr_t

Functions

- char * __FMT_osn_ip_addr (char *buf, size_t sz, const osn_ip_addr_t *addr)
- bool osn_ip_addr_from_str (osn_ip_addr_t *out, const char *str)
- bool osn_ip_addr_from_in_addr (osn_ip_addr_t *out, const struct in_addr *in)
- bool osn_ip_addr_from_sockaddr (osn_ip_addr_t *out, const struct sockaddr *in)
- int osn_ip_addr_cmp (void *a, void *b)
- osn_ip_addr_t osn_ip_addr_subnet (osn_ip_addr_t *addr)
- osn_ip_addr_t osn_ip_addr_to_bcast (osn_ip_addr_t *addr)
- int osn_ip_addr_to_prefix (osn_ip_addr_t *addr)
- osn_ip_addr_t osn_ip_addr_from_prefix (int prefix)
- char * __FMT_osn_ip6_addr (char *buf, size_t sz, const osn_ip6_addr_t *addr)
- bool osn_ip6_addr_from_str (osn_ip6_addr_t *out, const char *str)
- int osn_ip6_addr_cmp (void *a, void *b)
- int osn_ip6_addr_nolft_cmp (void *_a, void *_b)
- bool osn_mac_addr_from_str (osn_mac_addr_t *out, const char *str)
- int osn_mac_addr_cmp (void *_a, void *_b)

4.7.1 Detailed Description

OpenSync Networking Common Types.

4.8 osn_upnp.h File Reference

OpenSync UPnP.

```
#include "osn_types.h"
```

Typedefs

- typedef struct osn_upnp osn_upnp_t

Enumerations

- enum osn_upnp_mode {
  UPNP_MODE_NONE,
  UPNP_MODE_INTERNAL,
  UPNP_MODE_EXTERNAL }

Functions

- osn_upnp_t * osn_upnp_new (const char *ifname)
- bool osn_upnp_del (osn_upnp_t *self)
- bool osn_upnp_start (osn_upnp_t *self)
- bool osn_upnp_stop (osn_upnp_t *self)
- bool osn_upnp_set (osn_upnp_t *self, enum osn_upnp_mode mode)
- bool osn_upnp_get (osn_upnp_t *self, enum osn_upnp_mode *mode)

4.8.1 Detailed Description

OpenSync UPnP.

4.9 osn_vlan.h File Reference

OpenSync VLAN Interface Abstraction.

```
#include <stdbool.h>
#include "osn_types.h"
```

Typedefs

- typedef struct osn_vlan osn_vlan_t

Functions

- osn_vlan_t * osn_vlan_new (const char *ifname)
- bool osn_vlan_del (osn_vlan_t *self)
- bool osn_vlan_apply (osn_vlan_t *self)
- bool osn_vlan_parent_set (osn_vlan_t *self, const char *parent_ifname)
- bool osn_vlan_vid_set (osn_vlan_t *self, int vlanid)

4.9.1 Detailed Description

OpenSync VLAN Interface Abstraction.

4.10 osp.h File Reference

Platform APIs.

```
#include "osp_unit.h"
#include "osp_tm.h"
#include "osp_reboot.h"
#include "osp_led.h"
#include "osp_btn.h"
#include "osp_upg.h"
#include "osp_ps.h"
#include "osp_dl.h"
#include "osp_objm.h"
```

4.10.1 Detailed Description

Platform APIs.

Note

The individual headers are included here for backward compatibility and may be removed in the future. It is recommended to include just the needed header(s) in source files.

4.11 osp_btn.h File Reference

Button API.

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

Classes

- struct osp_btn_event

Typedefs

- typedef void(* osp_btn_cb) (void *obj, enum osp_btn_name name, const struct osp_btn_event *event)

Enumerations

- enum osp_btn_name {
  OSP_BTN_NAME_RESET = (1 << 0),
  OSP_BTN_NAME_WPS = (1 << 1) }

Functions

- bool osp_btn_get_caps (uint32_t *caps)
- bool osp_btn_register (osp_btn_cb cb, void *obj)

4.11.1 Detailed Description

Button API.

4.12 osp_dl.h File Reference

OSP Download API.

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

Typedefs

- typedef void(* osp_dl_cb) (const enum osp_dl_status status, void *cb_ctx)

Enumerations

- enum osp_dl_status {
  OSP_DL_OK = 0,
  OSP_DL_DOWNLOAD_FAILED,
  OSP_DL_ERROR }

Functions

- bool osp_dl_download (char *url, char *dst_path, int timeout, osp_dl_cb dl_cb, void *cb_ctx)

4.12.1 Detailed Description

OSP Download API.

4.13 osp_led.h File Reference

LED API.

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

Macros

- #define OSP_LED_PRIORITY_DISABLE ((uint32_t)-1)
- #define OSP_LED_PRIORITY_DEFAULT ((uint32_t)-2)

Enumerations

- enum osp_led_state {
  OSP_LED_ST_IDLE = 0,
  OSP_LED_ST_ERROR,
  OSP_LED_ST_CONNECTED,
  OSP_LED_ST_CONNECTING,
  OSP_LED_ST_CONNECTFAIL,
  OSP_LED_ST_WPS,
  OSP_LED_ST_OPTIMIZE,
  OSP_LED_ST_LOCATE,
  OSP_LED_ST_HWERROR,
  OSP_LED_ST_THERMAL,
  OSP_LED_ST_BTCONNECTING,
  OSP_LED_ST_BTCONNECTED,
  OSP_LED_ST_BTCONNECTFAIL,
  OSP_LED_ST_UPGRADING,
  OSP_LED_ST_UPGRADED,
  OSP_LED_ST_UPGRADEFAIL,
  OSP_LED_ST_HWTEST,
  OSP_LED_ST_LAST }

Functions

- int osp_led_init (int *led_cnt)
- int osp_led_set_state (enum osp_led_state state, uint32_t priority)
- int osp_led_clear_state (enum osp_led_state state)
- int osp_led_reset (void)
- int osp_led_get_state (enum osp_led_state *state, uint32_t *priority)
- const char * osp_led_state_to_str (enum osp_led_state state)
- enum osp_led_state osp_led_str_to_state (const char *str)

4.13.1 Detailed Description

LED API.

4.14 osp_objm.h File Reference

OSP Object Management API.

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

Functions

- bool osp_objm_install (char *path, char *name, char *version)
- bool osp_objm_remove (char *name, char *version)
- bool osp_objm_path (char *buf, size_t buffsz, char *name, char *version)

4.14.1 Detailed Description

OSP Object Management API.

4.15 osp_ps.h File Reference

Persistent Storage API.

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

Macros

- #define OSP_PS_READ (1 << 0)
- #define OSP_PS_WRITE (1 << 1)
- #define OSP_PS_PRESERVE (1 << 2)
- #define OSP_PS_RDWR (OSP_PS_READ | OSP_PS_WRITE)

Typedefs

- typedef struct osp_ps osp_ps_t

Functions

- osp_ps_t * osp_ps_open (const char *store, int flags)
- bool osp_ps_close (osp_ps_t *ps)
- ssize_t osp_ps_set (osp_ps_t *ps, const char *key, void *value, size_t value_sz)
- ssize_t osp_ps_get (osp_ps_t *ps, const char *key, void *value, size_t value_sz)
- bool osp_ps_erase (osp_ps_t *ps)
- bool osp_ps_sync (osp_ps_t *ps)

4.15.1 Detailed Description

Persistent Storage API.

4.16 osp_reboot.h File Reference

Reboot API.

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

Enumerations

- enum osp_reboot_type {
  OSP_REBOOT_UNKNOWN,
  OSP_REBOOT_COLD_BOOT,
  OSP_REBOOT_POWER_CYCLE,
  OSP_REBOOT_WATCHDOG,
  OSP_REBOOT_CRASH,
  OSP_REBOOT_USER,
  OSP_REBOOT_DEVICE,
  OSP_REBOOT_HEALTH_CHECK,
  OSP_REBOOT_UPGRADE,
  OSP_REBOOT_THERMAL,
  OSP_REBOOT_CLOUD,
  OSP_REBOOT_CANCEL }

Functions

- bool osp_unit_reboot_ex (enum osp_reboot_type type, const char ∗reason, int ms_delay)
- bool osp_unit_factory_reboot (const char ∗reason, int ms_delay)
- bool osp_unit_reboot_get (enum osp_reboot_type ∗type, char ∗reason, ssize_t reason_sz)

4.16.1 Detailed Description

Reboot API.

4.17 osp_tm.h File Reference

Thermal Management API.

```
#include <stdbool.h>
```

Classes

- struct osp_tm_therm_state

Macros

- #define OSP_TM_TEMP_SRC_MAX (3)
- #define OSP_TM_TEMP_AVG_CNT (3)

Functions

- int osp_tm_init (const struct osp_tm_therm_state **tbl, unsigned int *therm_state_cnt, unsigned int *temp_↩ src_cnt, void **priv)
- void osp_tm_deinit (void *priv)
- bool osp_tm_is_temp_src_enabled (void *priv, int idx)
- const char * osp_tm_get_temp_src_name (void *priv, int idx)
- int osp_tm_get_temperature (void *priv, int idx, int *temp)
- int osp_tm_get_fan_rpm (void *priv, unsigned int *rpm)
- int osp_tm_set_fan_rpm (void *priv, unsigned int rpm)

4.17.1 Detailed Description

Thermal Management API.

4.18 osp_unit.h File Reference

OSP Unit API.

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

Functions

- bool osp_unit_id_get (char *buff, size_t buffsz)

  Return device identification.
- bool osp_unit_serial_get (char *buff, size_t buffsz)

  Return device serial number.
- bool osp_unit_model_get (char *buff, size_t buffsz)

  Return device model.
- bool osp_unit_sku_get (char *buff, size_t buffsz)

  Return device stock keeping unit number.
- bool osp_unit_hw_revision_get (char *buff, size_t buffsz)

  Return hardware version number.
- bool osp_unit_platform_version_get (char *buff, size_t buffsz)

  Return platform version number.
- bool osp_unit_sw_version_get (char *buff, size_t buffsz)

Return software version number.

- bool osp_unit_vendor_name_get (char *buff, size_t buffsz)

  Return vendor name.
- bool osp_unit_vendor_part_get (char *buff, size_t buffsz)

  Return vendor part number.
- bool osp_unit_manufacturer_get (char *buff, size_t buffsz)

  Return manufacturer name.
- bool osp_unit_factory_get (char *buff, size_t buffsz)

  Return factory name.
- bool osp_unit_mfg_date_get (char *buff, size_t buffsz)

  Return manufacturing date.

4.18.1 Detailed Description

OSP Unit API.

4.19 osp_upg.h File Reference

Upgrade API.

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

Typedefs

- typedef void(* osp_upg_cb) (const osp_upg_op_t op, const osp_upg_status_t status, uint8_t completed)

Enumerations

- enum osp_upg_op_t {
  OSP_UPG_DL,
  OSP_UPG_UPG }
- enum osp_upg_status_t {
  OSP_UPG_OK = 0,
  OSP_UPG_ARGS = 1,
  OSP_UPG_URL = 3,
  OSP_UPG_DL_FW = 4,
  OSP_UPG_DL_MD5 = 5,
  OSP_UPG_MD5_FAIL = 6,
  OSP_UPG_IMG_FAIL = 7,
  OSP_UPG_FL_ERASE = 8,
  OSP_UPG_FL_WRITE = 9,
  OSP_UPG_FL_CHECK = 10,
  OSP_UPG_BC_SET = 11,
  OSP_UPG_APPLY = 12,
  OSP_UPG_BC_ERASE = 14,
  OSP_UPG_SU_RUN = 15,
  OSP_UPG_DL_NOFREE = 16,
  OSP_UPG_WRONG_PARAM = 17,
  OSP_UPG_INTERNAL = 18 }

Functions

- bool osp_upg_check_system (void)
- bool osp_upg_dl (char *url, uint32_t timeout, osp_upg_cb dl_cb)
- bool osp_upg_upgrade (char *password, osp_upg_cb upg_cb)
- bool osp_upg_commit (void)
- int osp_upg_errno (void)

4.19.1 Detailed Description

Upgrade API.

4.20 target.h File Reference

Base target API header.

```
#include <stdbool.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <ev.h>
#include "os.h"
#include "os_types.h"
#include "schema.h"
#include "os_backtrace.h"
#include "target_bsal.h"
```

Classes

- struct target_managers_config_t

Macros

- #define **TARGET_BUFF_SZ** 256
- #define **TARGET_SERIAL_SZ** OS_MACSTR_PLAIN_SZ
- #define **TARGET_ID_SZ** OS_MACSTR_PLAIN_SZ

Enumerations

- enum **target_init_opt_t** {
  TARGET_INIT_COMMON = 0,
  TARGET_INIT_MGR_DM = 1,
  TARGET_INIT_MGR_CM = 2,
  TARGET_INIT_MGR_WM = 3,
  TARGET_INIT_MGR_SM = 4,
  TARGET_INIT_MGR_NM = 5,
  TARGET_INIT_MGR_BM = 6,
  TARGET_INIT_MGR_FM = 7,
  TARGET_INIT_MGR_LM = 8,
  TARGET_INIT_MGR_LEDM = 9,
  TARGET_INIT_MGR_OM = 10,
  TARGET_INIT_MGR_BLEM = 11,
  TARGET_INIT_MGR_QM = 12,
  TARGET_INIT_MGR_PM = 13,
  TARGET_INIT_MGR_FSM = 14,
  TARGET_INIT_MGR_TM = 15,
  TARGET_INIT_MGR_HELLO_WORLD = 16,
  TARGET_INIT_MGR_FCM = 17,
  TARGET_INIT_MGR_PPM = 18,
  TARGET_INIT_MGR_NFM = 19 }

Functions

- bool target_ready (struct ev_loop *loop)

  Wait for platform readiness.
- bool target_init (target_init_opt_t opt, struct ev_loop *loop)

  Perform platform specific initialization.
- bool target_close (target_init_opt_t opt, struct ev_loop *loop)

  Perform platform specific cleanups on exit.
- bool target_is_radio_interface_ready (char *phy_name)

  Check if radio interface is ready.
- bool target_is_interface_ready (char *if_name)

  Check if interface is ready.
- const char * target_wan_interface_name ()

  Get wan interface name.
- const char ** **target_ethclient_iflist_get** ()
- const char ** **target_ethclient_brlist_get** ()
- bool **target_map_init** (void)
- bool **target_map_close** (void)
- bool **target_map_insert** (char *if_name, char *map_name)
- char * **target_map_ifname** (char *ifname)
- bool **target_map_ifname_exists** (const char *ifname)
- char * **target_unmap_ifname** (char *ifname)
- bool **target_unmap_ifname_exists** (const char *ifname)
- const char * target_tls_cacert_filename (void)

  Get the TLS CA certificate filename.
- const char * target_tls_mycert_filename (void)

  Get the TLS certificate filename.

- const char * target_tls_privkey_filename (void)

  Get the TLS private key filename.
- bool target_log_open (char *name, int flags)

  Enable logging.
- bool target_log_pull (const char *upload_location, const char *upload_token)

  Collect logs.
- bool target_log_pull_ext (const char *upload_location, const char *upload_token, const char *upload_method)

  Collect logs (using specified method).
- INTERNAL const char * target_scripts_dir (void)

  Get the directory for scripts.
- const char * target_tools_dir (void)

  Get the directory for tools.
- const char * target_bin_dir (void)

  Get the directory for binaries.
- const char * target_persistent_storage_dir (void)

  Get a persistent storage mount point.
- INTERNAL void target_managers_restart (void)

  Restart all managers.

Variables

- target_managers_config_t target_managers_config [ ]

  List of managers to start.
- int **target_managers_num**

4.20.1 Detailed Description

Base target API header.

At the end of this header the platform specific header TARGET_H is also included, which can declare additional APIs and will usually include also the target_common.h

4.21 target_bsal.h File Reference

Band Steering target API subset.

Classes

- struct bsal_ifconfig_t
- struct bsal_client_config_t
- struct bsal_neigh_info_t
- struct bsal_btm_params_t
- struct bsal_rrm_params_t
- struct bsal_datarate_info_t
- struct bsal_rrm_caps_t
- struct bsal_ev_probe_req_t
- struct bsal_ev_connect_t
- struct bsal_ev_disconnect_t
- struct bsal_ev_activity_t
- struct bsal_ev_chan_util_t
- struct bsal_ev_rssi_xing_t
- struct bsal_ev_rssi_t
- struct bsal_ev_steer_t
- struct bsal_ev_auth_fail_t
- struct bsal_ev_action_frame_t
- struct bsal_event_t
- struct bsal_client_info_t

Macros

- #define **BSAL_IFNAME_LEN** 17
- #define **BSAL_MAC_ADDR_LEN** 6
- #define **BSAL_MAX_TM_NEIGHBORS** 3
- #define **BSAL_MAX_ASSOC_IES_LEN** 1024
- #define **BSAL_MAX_ACTION_FRAME_LEN** 1024

Typedefs

- typedef void(∗ **bsal_event_cb_t**) (bsal_event_t ∗event)

Enumerations

- enum bsal_ev_type_t {
  BSAL_EVENT_PROBE_REQ = 1,
  BSAL_EVENT_CLIENT_CONNECT,
  BSAL_EVENT_CLIENT_DISCONNECT,
  BSAL_EVENT_CLIENT_ACTIVITY,
  BSAL_EVENT_CHAN_UTILIZATION,
  BSAL_EVENT_RSSI_XING,
  BSAL_EVENT_RSSI,
  BSAL_EVENT_STEER_CLIENT,
  BSAL_EVENT_STEER_SUCCESS,
  BSAL_EVENT_STEER_FAILURE,
  BSAL_EVENT_AUTH_FAIL,
  BSAL_EVENT_ACTION_FRAME,
  BSAL_EVENT_DEBUG_CHAN_UTIL = 128,
  BSAL_EVENT_DEBUG_RSSI }

- enum **bsal_disc_source_t** {
  BSAL_DISC_SOURCE_LOCAL = 0,
  BSAL_DISC_SOURCE_REMOTE }
- enum **bsal_disc_type_t** {
  BSAL_DISC_TYPE_DISASSOC = 0,
  BSAL_DISC_TYPE_DEAUTH }
- enum **bsal_rssi_change_t** {
  BSAL_RSSI_UNCHANGED = 0,
  BSAL_RSSI_HIGHER,
  BSAL_RSSI_LOWER }
- enum **bsal_phy_mode_t** {
  BSAL_PHY_MODE_AUTO = 0,
  BSAL_PHY_MODE_11A = 1,
  BSAL_PHY_MODE_11B = 2,
  BSAL_PHY_MODE_11G = 3,
  BSAL_PHY_MODE_FH = 4,
  BSAL_PHY_MODE_TURBO_A = 5,
  BSAL_PHY_MODE_TURBO_G = 6,
  BSAL_PHY_MODE_11NA_HT20 = 7,
  BSAL_PHY_MODE_11NG_HT20 = 8,
  BSAL_PHY_MODE_11NA_HT40PLUS = 9,
  BSAL_PHY_MODE_11NA_HT40MINUS = 10,
  BSAL_PHY_MODE_11NG_HT40PLUS = 11,
  BSAL_PHY_MODE_11NG_HT40MINUS = 12,
  BSAL_PHY_MODE_11NG_HT40 = 13,
  BSAL_PHY_MODE_11NA_HT40 = 14,
  BSAL_PHY_MODE_11AC_VHT20 = 15,
  BSAL_PHY_MODE_11AC_VHT40PLUS = 16,
  BSAL_PHY_MODE_11AC_VHT40MINUS = 17,
  BSAL_PHY_MODE_11AC_VHT40 = 18,
  BSAL_PHY_MODE_11AC_VHT80 = 19,
  BSAL_PHY_MODE_11AC_VHT160 = 20,
  BSAL_PHY_MODE_11AC_VHT80_80 = 21 }
- enum **bsal_max_chwidth_t** {
  BSAL_MAX_CHWIDTH_20MHZ = 0,
  BSAL_MAX_CHWIDTH_40MHZ = 1,
  BSAL_MAX_CHWIDTH_80MHZ = 2,
  BSAL_MAX_CHWIDTH_160MHZ = 3 }

Functions

- int target_bsal_init (bsal_event_cb_t event_cb, struct ev_loop ∗loop)

  Gives target a chance to hook and initialize ∗ internals.
- int target_bsal_cleanup (void)

  Gives target a chance to clean up on shutdown.
- int target_bsal_iface_add (const bsal_ifconfig_t ∗ifcfg)

  Requests target to start managing provided interface.
- int target_bsal_iface_update (const bsal_ifconfig_t ∗ifcfg)

  Requests target to update configuration on already managed interface.
- int target_bsal_iface_remove (const bsal_ifconfig_t ∗ifcfg)

  Requests target to stop managing provided interface.
- int target_bsal_client_add (const char ∗ifname, const uint8_t ∗mac_addr, const bsal_client_config_t ∗conf)

Requests target to start managing provided client.

- int target_bsal_client_update (const char *ifname, const uint8_t *mac_addr, const bsal_client_config_t *conf)

  Requests target to update provided client policy configuration.
- int target_bsal_client_remove (const char *ifname, const uint8_t *mac_addr)

  Requests target to stop managing a client.
- int target_bsal_client_measure (const char *ifname, const uint8_t *mac_addr, int num_samples)

  Requests target to schedule signal strength measurement.
- int target_bsal_client_disconnect (const char *ifname, const uint8_t *mac_addr, bsal_disc_type_t type, uint8_t reason)

  Requests target to disconnect a client.
- int target_bsal_client_info (const char *ifname, const uint8_t *mac_addr, bsal_client_info_t *info)

  Requests target to provide client capabilities.
- int target_bsal_bss_tm_request (const char *ifname, const uint8_t *mac_addr, const bsal_btm_params_t *btm↩_params)

  Requests target to send a BSS Transition Request frame.
- int target_bsal_rrm_beacon_report_request (const char *ifname, const uint8_t *mac_addr, const bsal_rrm_↩params_t *rrm_params)

  Requests target to send RRM Beacon Report Request frame.
- int target_bsal_rrm_set_neighbor (const char *ifname, const bsal_neigh_info_t *nr)

  Requests target to add an entry to neighbor list.
- int target_bsal_rrm_remove_neighbor (const char *ifname, const bsal_neigh_info_t *nr)

  Requests target to remove an entry from neighbor list.
- int target_bsal_send_action (const char *ifname, const uint8_t *mac_addr, const uint8_t *data, unsigned int data_len)

  Request target to send action frame.

4.21.1 Detailed Description

Band Steering target API subset.

This is mostly used by BM (Band Steering Manager) to interact with the target for the purpose of performing Wireless client steering.

The list of managed clients doesn't tend to change during runtime a lot. Cloud will often program list of all home AP interfaces and all clients it has ever seen in a location upon onboarding.

It then may end up adding new client entries if never seen before clients show up. It's more often expected to see a client policy being changed in runtime.

4.22 target_common.h File Reference

Additional target API header.

```
#include "dppline.h"
#include "ds_dlist.h"
#include "schema.h"
```

Classes

- struct target_radio_ops

  List of callbacks for radio/vif changes. *More...*
- struct target_connectivity_check_t
- struct mcproxyd_params

Macros

- #define TARGET_GW_TYPE (1 << 0)
- #define TARGET_EXTENDER_TYPE (1 << 1)

Typedefs

- typedef bool **target_stats_clients_cb_t**(ds_dlist_t *client_list, void *ctx, int status)
- typedef bool **target_stats_survey_cb_t**(ds_dlist_t *survey_list, void *survey_ctx, int status)
- typedef bool **target_scan_cb_t**(void *scan_ctx, int status)
- typedef bool target_mac_learning_cb_t(struct schema_OVS_MAC_Learning *omac, bool oper_status)

  Ethernet client change callback type.
- typedef char **ifname**[64]
- typedef struct mcproxyd_params target_mcproxy_params_t
- typedef bool **target_client_nickname_cb_t**(struct schema_Client_Nickname_Config *cncfg, bool status)
- typedef bool **target_client_freeze_cb_t**(struct schema_Client_Freeze_Config *cfcfg, bool status)

Enumerations

- enum target_connectivity_check_option_t {
  LINK_CHECK = 1 << 0,
  ROUTER_CHECK = 1 << 1,
  INTERNET_CHECK = 1 << 2,
  NTP_CHECK = 1 << 3 }
- enum target_prtcl_t {
  DISABLE_IGMP = 1,
  DISABLE_MLD,
  IGMPv1,
  IGMPv2,
  IGMPv3,
  MLDv1,
  MLDv2 }

Functions

- bool target_radio_init (const struct target_radio_ops *ops)

  Hands over WM callbacks so target can notify about vif/radio statuses.
- bool target_radio_config_init2 (void)

  Initialize radio interface configuration.
- bool target_radio_config_need_reset (void)

  Target tells if it requires full re-sync with Config/State.
- bool target_radio_config_set2 (const struct schema_Wifi_Radio_Config *rconf, const struct schema_Wifi_↩ Radio_Config_flags *changed)

  Apply the configuration for the radio interface.
- bool target_radio_state_get (char *ifname, struct schema_Wifi_Radio_State *rstate)

  Get state of radio interface.
- bool target_vif_config_set2 (const struct schema_Wifi_VIF_Config *vconf, const struct schema_Wifi_Radio_↩ Config *rconf, const struct schema_Wifi_Credential_Config *cconfs, const struct schema_Wifi_VIF_Config_flags *changed, int num_cconfs)

  Apply the configuration for the vif interface.
- bool target_vif_state_get (char *ifname, struct schema_Wifi_VIF_State *vstate)

  Get state of vif interface.
- bool target_radio_tx_stats_enable (radio_entry_t *radio_cfg, bool status)

  Enable radio tx stats.
- bool target_radio_fast_scan_enable (radio_entry_t *radio_cfg, ifname_t if_name)

  Enable radio fast scan.
- target_client_record_t * **target_client_record_alloc** ()
- void **target_client_record_free** (target_client_record_t *record)
- bool target_stats_clients_get (radio_entry_t *radio_cfg, radio_essid_t *essid, target_stats_clients_cb_t *client↩ _cb, ds_dlist_t *client_list, void *client_ctx)

  Get clients stats.
- bool target_stats_clients_convert (radio_entry_t *radio_cfg, target_client_record_t *client_list_new, target_↩ client_record_t *client_list_old, dpp_client_record_t *client_record)

  Calculate client stats deltas.
- target_survey_record_t * **target_survey_record_alloc** ()
- void **target_survey_record_free** (target_survey_record_t *record)
- bool target_stats_survey_get (radio_entry_t *radio_cfg, uint32_t *chan_list, uint32_t chan_num, radio_scan_↩ type_t scan_type, target_stats_survey_cb_t *survey_cb, ds_dlist_t *survey_list, void *survey_ctx)

  Get radio channel survey stats.
- bool target_stats_survey_convert (radio_entry_t *radio_cfg, radio_scan_type_t scan_type, target_survey_↩ record_t *data_new, target_survey_record_t *data_old, dpp_survey_record_t *survey_record)

  Calculate channel survey deltas.
- bool target_stats_scan_start (radio_entry_t *radio_cfg, uint32_t *chan_list, uint32_t chan_num, radio_scan_↩ type_t scan_type, int32_t dwell_time, target_scan_cb_t *scan_cb, void *scan_ctx)

  Start neighbor scan.
- bool target_stats_scan_stop (radio_entry_t *radio_cfg, radio_scan_type_t scan_type)

  Stop neighbor scan.
- bool target_stats_scan_get (radio_entry_t *radio_cfg, uint32_t *chan_list, uint32_t chan_num, radio_scan_↩ type_t scan_type, dpp_neighbor_report_data_t *scan_results)

  Get neighbor stats.
- bool target_stats_device_get (dpp_device_record_t *device_entry)

  Get device stats.
- bool target_stats_device_temp_get (radio_entry_t *radio_cfg, dpp_device_temp_t *device_entry)

Get device temperature.

- bool target_stats_device_txchainmask_get (radio_entry_t *radio_cfg, dpp_device_txchainmask_t *txchainmask_entry)

  Get device txchainmask.
- bool target_stats_device_fanrpm_get (uint32_t *fan_rpm)

  Get device fan RPM.
- bool target_device_config_register (void *awlan_cb)

  Subscribe to changes of device config.
- bool target_device_config_set (struct schema_AWLAN_Node *awlan)

  Apply device config.
- bool target_device_execute (const char *cmd)

  Execute external tools.
- int target_device_capabilities_get ()

  Get device capabilities.
- bool target_device_connectivity_check (const char *ifname, target_connectivity_check_t *cstate, target_connectivity_check_option_t opts)

  Get device connectivity status.
- bool target_device_restart_managers ()

  Restart plume managers.
- bool target_device_wdt_ping ()

  Ping watchdog system.
- bool target_mac_learning_register (target_mac_learning_cb_t *omac_cb)

  Subscribe to ethernet client change events.
- bool target_set_igmp_mcproxy_params (target_mcproxy_params_t *mcparams)

  Applies config to mcproxy and reloads the corresponding daemon.
- bool target_get_igmp_mcproxy_params (target_mcproxy_params_t *mcparams)

  Get config from the mcproxy.
- bool target_set_mld_mcproxy_params (target_mcproxy_params_t *mcparams)

  Applies config to mcproxy and reloads the corresponding daemon.
- bool target_get_mld_mcproxy_params (target_mcproxy_params_t *mcparams)

  Get config from the mcproxy.
- bool target_set_igmp_mcproxy_sys_params (struct schema_IGMP_Config *iccfg)

  Applies mcproxy system parameters and reloads the corresponding proxy daemon.
- bool target_get_igmp_mcproxy_sys_params (struct schema_IGMP_Config *iccfg)

  Get mcproxy system parameters.
- bool target_set_mld_mcproxy_sys_params (struct schema_MLD_Config *mlcfg)

  Applies mcproxy system parameters and reloads the corresponding proxy daemon.
- bool target_get_mld_mcproxy_sys_params (struct schema_MLD_Config *iccfg)

  Get mcproxy system parameters.
- bool **target_client_nickname_register** (target_client_nickname_cb_t *nick_cb)
- bool **target_client_nickname_set** (struct schema_Client_Nickname_Config *cncfg)
- bool **target_client_freeze_register** (target_client_freeze_cb_t *freze_cb)
- bool **target_client_freeze_set** (struct schema_Client_Freeze_Config *cfcfg)

4.22.1 Detailed Description

Additional target API header.

The declarations in this header depend on the platform specific declaration from header TARGET_H, which is why it is separated from target.h

Index