Kinetis Thread Stack API Reference Manual

KTSAPIRM

Rev. 4
Jan 2018

Contents

Chapter 1
Thread Application Configuration Interface

Chapter 2
Thread Network Interface

Chapter 4
Thread Application Callbacks Interface

Chapter 7
Network IP Sockets Interface

Chapter 8
CoAP Interface

Chapter 9
Network IP Interface

Chapter 10
Thread Network Utilities Interface

Chapter 1
Thread Application Configuration Interface

1.1 Overview

Files

- file app_stack_config.h
- file app_thread_config.h
- file thread_cfg.h

Macros

- #define THREAD_USE_SHELL
- #define THREAD_USE_THCI
- #define THR_MAX_REED_ROUTERS_NEIGHBORS
- #define THR_MAX_SLEEPY_ED_NEIGHBORS
- #define THR_MAX_NEIGHBORS
- #define THR_MAX_DATA_REQS
- #define THR_FAILED_CHILD_TRANSMISSIONS
- #define THR_FAILED_ROUTER_TRANSMISSIONS
- #define DHCP6_SERVER_MAX_INSTANCES
- #define DHCP6_SERVER_MAX_CLIENTS
- #define DHCP6_CLIENT_MAX_INSTANCES
- #define COAP_MAX_SESSIONS
- #define BSDS_MAX_SOCKETS
- #define MAX_UDP_CONNECTIONS
- #define IP_IP6_ROUTING_TBL_SIZE
- #define IP_IP6_FIREWALL_TBL_SIZE
- #define IP_IF_NB
- #define IP_IF_IP6_ADDR_NB
- #define IP_IF_IP6_MULTICAST_ADDR_NB
- #define IP_TRANSPORT_SERVICE_NB
- #define IP_IP_REASSEMBLY_QUEUE_SIZE
- #define IP_IF_IP4_ADDR_NB
- #define MPL_INSTANCE_SET_SIZE
- #define MPL_SEED_SET_SIZE
- #define MPL_BUFFERED_MESSAGE_SET_SIZE
- #define TRICKLE_INSTANCE_SET_SIZE
- #define TRICKLE_LIST_SIZE
- #define SLWPCFG_INSTANCES_NB
- #define SLWPCFG_RFC6282_CONTEXT_TABLE_SIZE
- #define SLWPCFG_UNFRAG_SED_TRACK_NB
- #define SLWPCFG_UNFRAG_SED_TRACK_PKT_NB
- #define SLWPCFG_SED_IND_QUEUE_SIZE
- #define MAC_FILTERING_ENABLED
- #define MAC_FILTERING_TABLE_SIZE
- #define ThrPoolId_d
- #define THREAD_TASK_MSG_QUEUE_SIZE

- #define THREAD_TASK_STACK_SIZE
- #define THR_MAX_INSTANCES
- #define DEBUG_REED_AUTO_PROMOTE
- #define THR_SERVER_DATA_PREFIX_TBL_SIZE
- #define THR_SERVER_DATA_BR_SET_TBL_SIZE
- #define THR_SERVER_DATA_HAS_ROUTE_TBL_SIZE
- #define THR_LOCAL_SERVICE_SET_TBL_SIZE
- #define THR_NWK_DATA_SERVICE_SET_TBL_SIZE
- #define THR_SLAAC_TEMP_ADDR_TABLE_SIZE
- #define THR_NWK_DATA_PREFIX_TBL_SIZE
- #define THR_NWK_DATA_CTX_TBL_SIZE
- #define THR_NWK_DATA_BR_SET_TBL_SIZE
- #define THR_NWK_DATA_HAS_ROUTE_TBL_SIZE
- #define THR_NWK_DATA_MIN_STABLE_LIFETIME_SEC
- #define THR_LEADER_ID_SEQUENCE_PERIOD_SEC
- #define THR_CHILD_ADDR_REG_ENTIRES
- #define THR_CHILD_MCAST_ADDR_REG_ENTIRES
- #define THR_MAX_LINK_SYNC_NEIGHBORS
- #define THR_MAX_NWK_ATTACH_PARENT_ENTRIES
- #define THR_REATTACH_JITTER_MIN_MS
- #define THR_REATTACH_JITTER_MAX_MS
- #define THR_LEADER_TIMEOUT_SEC
- #define THR_MAX_ROUTERS
- #define THR_ROUTER_UPGRADE_THRESHOLD
- #define THR_ROUTER_DOWNGRADE_THRESHOLD
- #define THR_MIN_DOWNGRADE_NEIGHBORS
- #define THR_ROUTER_SELECTION_JITTER_SEC
- #define THR_MAX_DEV_ADDR_QUERY_CACHE_ENTRIES
- #define THR_ADDRESS_QUERY_TIMEOUT_SEC
- #define THR_ADDRESS_QUERY_INITIAL_RETRY_DELAY_SEC
- #define THR_ADDRESS_QUERY_MAX_RETRY_DELAY_SEC
- #define THR_POWERON_ROUTER_MIN_JITTER_MS
- #define THR_POWERON_ROUTER_MAX_JITTER_MS
- #define THR_POWERON_ED_MAX_JITTER_MS
- #define THR_PARENT_ROUTE_TO_LEADER_TIMEOUT_MS
- #define THR_CHILD_ED_KEEP_ALIVE_INTERVAL_MIN_MS
- #define THR_CHILD_ED_KEEP_ALIVE_INTERVAL_MAX_MS
- #define THR_CONTEXT_REUSE_DELAY_SEC
- #define THR_DISCOVERY_EXT_ADDR
- #define THR_DISCOVERY_KEY
- #define THR_DISCOVERY_FRAME_COUNTER
- #define THR_DISCOVERY_TIME
- #define THR_DISCOVERY_MAX_JITTER

1.2 Macro Definition Documentation

1.2.1 #define THREAD_USE_SHELL

Thread APP uses SHELL commands.

1.2.2 #define THREAD_USE_THCI

Thread APP uses FCSI commands.

1.2.3 #define THR_MAX_REED_ROUTERS_NEIGHBORS

The maximum number of Thread Router / REED (Router Eligible Devices) radio range neighbors.

1.2.4 #define THR_MAX_SLEEPY_ED_NEIGHBORS

The maximum number of Thread Sleepy End Device radio range neighbors.

1.2.5 #define THR_MAX_NEIGHBORS

The maximum number of radio range neighbors with which the Thread device can communicate.

1.2.6 #define THR_MAX_DATA_REQS

The maximum number of simultaneous 802.15.4 transmissions.

1.2.7 #define THR_FAILED_CHILD_TRANSMISSIONS

Number of consecutive failed 802.15.4 transmissions for a link to be considered down and a child device to reattach.

1.2.8 #define THR_FAILED_ROUTER_TRANSMISSIONS

Number of consecutive failed 802.15.4 transmissions for a Router-to-Router link to be considered broken.

1.2.9 #define DHCP6_SERVER_MAX_INSTANCES

The maximum number of DHCPv6 servers that can be started on the device.

1.2.10 #define DHCP6_SERVER_MAX_CLIENTS

The maximum number of DHCPv6 clients that the device can service as a DHCPv6 server.

1.2.11 #define DHCP6_CLIENT_MAX_INSTANCES

The maximum number of DHCPv6 clients that can be started on the device.

1.2.12 #define COAP_MAX_SESSIONS

The maximum number of COAP sessions that can be established at one time.

1.2.13 #define BSDS_MAX_SOCKETS

The maximum number of sockets that can be opened at one time.
MUST be correlated to MAX_UDP_CONNECTIONS

1.2.14 #define MAX_UDP_CONNECTIONS

The maximum number of UDP connections that can be opened at one time.
MUST not be greater than BSDS_MAX_SOCKETS

1.2.15 #define IP_IP6_ROUTING_TBL_SIZE

The maximum number of IP route entries.

1.2.16 #define IP_IP6_FIREWALL_TBL_SIZE

The maximum number of dynamic firewall entries.

1.2.17 #define IP_IF_NB

The maximum supported number of IP interfaces.

1.2.18 #define IP_IF_IP6_ADDR_NB

The maximum number of IPv6 addresses.
This is regardless of how many interfaces are available

1.2.19 #define IP_IF_IP6_MULTICAST_ADDR_NB

The maximum number of supported multicast addresses.

1.2.20 #define IP_TRANSPORT_SERVICE_NB

The maximum number of IP transport services that can be supported.

Ex. UDP, TCP.

1.2.21 #define IP_IP_REASSEMBLY_QUEUE_SIZE

Number representing how many IP packet fragments can be stored at one time.

1.2.22 #define IP_IF_IP4_ADDR_NB

The maximum number of IPv4 addresses.

This is regardless of how many interfaces are available

1.2.23 #define MPL_INSTANCE_SET_SIZE

The maximum number of MPL instances.

This must be correlated to IP_IF_NB.

1.2.24 #define MPL_SEED_SET_SIZE

The maximum number of seeds the MPL module can store at one time.

1.2.25 #define MPL_BUFFERED_MESSAGE_SET_SIZE

The maximum number of MPL transmitted messages that can be buffered at one time.

1.2.26 #define TRICKLE_INSTANCE_SET_SIZE

The maximum number of TRICKLE instances.

This must be correlated to IP_IF_NB

1.2.27 #define TRICKLE_LIST_SIZE

The maximum number of Trickle events.

1.2.28 #define SLWPCFG_INSTANCES_NB

The maximum number of 6LoWPAN instances.
MUST not be greater than IP_IF_NB

1.2.29 #define SLWPCFG_RFC6282_CONTEXT_TABLE_SIZE

The maximum number of 6LoWPAN contexts that can be stored.

1.2.30 #define SLWPCFG_UNFRAG_SED_TRACK_NB

The number of SED devices a router can handle for unfragmented packets.

1.2.31 #define SLWPCFG_UNFRAG_SED_TRACK_PKT_NB

The number of unfragmented packets a parent can hold for a SED.

1.2.32 #define SLWPCFG_SED_IND_QUEUE_SIZE

The number of SED fragmented packets a parent can hold for transmission.

1.2.33 #define MAC_FILTERING_ENABLED

Enables/Disables the MAC Filtering.

1.2.34 #define MAC_FILTERING_TABLE_SIZE

The maximum number of entries in the MAC filtering table.

1.2.35 #define ThrPoolId_d

The message pool ID used for thread stack.

1.2.36 #define THREAD_TASK_MSG_QUEUE_SIZE

The size of the massage queue used by Thread task.

1.2.37 #define THREAD_TASK_STACK_SIZE

The stack size of Thread task.

1.2.38 #define THR_MAX_INSTANCES

The maximum number of Thread Interfaces.
MUST not be greater that IP_IF_NB

1.2.39 #define DEBUG_REED_AUTO_PROMOTE

Debug flag for auto promote.

1.2.40 #define THR_SERVER_DATA_PREFIX_TBL_SIZE

The size of the Server Data prefix table.

1.2.41 #define THR_SERVER_DATA_BR_SET_TBL_SIZE

The size of Border Route table for local Server Data.

1.2.42 #define THR_SERVER_DATA_HAS_ROUTE_TBL_SIZE

The size of Has Route table for local Server Data.

1.2.43 #define THR_LOCAL_SERVICE_SET_TBL_SIZE

The size of local BR service set.

1.2.44 #define THR_NWK_DATA_SERVICE_SET_TBL_SIZE

The size of Nwk Data Service Set table.

1.2.45 #define THR_SLAAC_TEMP_ADDR_TABLE_SIZE

The size of Thread Slaac temporary address table - stored in NVM (NotMirroredInRam)

1.2.46 #define THR_NWK_DATA_PREFIX_TBL_SIZE

The size of NWK Data prefix table.

1.2.47 #define THR_NWK_DATA_CTX_TBL_SIZE

The size of NWK Data context table.

1.2.48 #define THR_NWK_DATA_BR_SET_TBL_SIZE

The size of NWK Data Border Set table.

1.2.49 #define THR_NWK_DATA_HAS_ROUTE_TBL_SIZE

The size of NWK Data Has Route table.

1.2.50 #define THR_NWK_DATA_MIN_STABLE_LIFETIME_SEC

The lifetime for stable NWK Data.

1.2.51 #define THR_LEADER_ID_SEQUENCE_PERIOD_SEC

The maximum interval between increments of ID_sequence_number by the Leader.

1.2.52 #define THR_CHILD_ADDR_REG_ENTIRES

The number of entries in the Address registration table per RFD child.

1.2.53 #define THR_CHILD_MCAST_ADDR_REG_ENTIRES

The number of entries in the Multicast Address registration table per RFD child.

1.2.54 #define THR_MAX_LINK_SYNC_NEIGHBORS

The max number of neighbors to do a link sync.

1.2.55 #define THR_MAX_NWK_ATTACH_PARENT_ENTRIES

The maximum number of parents selected to attach with.

1.2.56 #define THR_REATTACH_JITTER_MIN_MS

The minimum jitter time for generating the random period used at re-attaching the device.

1.2.57 #define THR_REATTACH_JITTER_MAX_MS

The maximum jitter time for generating the random period used at re-attaching the device.

1.2.58 #define THR_LEADER_TIMEOUT_SEC

The maximum number of seconds for a Router to get disconnected from the Leader if no ID_sequence_↩ number is received from a neighbor.

If a Router goes for THR_LEADER_TIMEOUT_SEC seconds without receiving a new ID_sequence_↩ number from a neighbor, it MUST consider itself disconnected from the Leader and stop using its current Router ID

1.2.59 #define THR_MAX_ROUTERS

The maximum number of allowed Routers in the Thread network.

Maximum value can be 32

1.2.60 #define THR_ROUTER_UPGRADE_THRESHOLD

The number of active Routers on the Thread Network below which a REED may decide to become a Router.

1.2.61 #define THR_ROUTER_DOWNGRADE_THRESHOLD

The number of active Routers on the Thread Network above which an active Router may decide to become a Child.

1.2.62 #define THR_MIN_DOWNGRADE_NEIGHBORS

The minimum number of neighbors with link quality 2 or better that a Router must have to downgrade to a REED.

It should be less than 32

1.2.63 #define THR_ROUTER_SELECTION_JITTER_SEC

The maximum jitter time when soliciting a router ID.

1.2.64 #define THR_MAX_DEV_ADDR_QUERY_CACHE_ENTRIES

The max number of cache entries for address query.

1.2.65 #define THR_ADDRESS_QUERY_TIMEOUT_SEC

The time needed for an address query to complete.

1.2.66 #define THR_ADDRESS_QUERY_INITIAL_RETRY_DELAY_SEC

The minimum delay between 2 address queries.

1.2.67 #define THR_ADDRESS_QUERY_MAX_RETRY_DELAY_SEC

The maximum delay betwenn 2 address queries.

1.2.68 #define THR_POWERON_ROUTER_MIN_JITTER_MS

On power on, during the network start with NVM, a router will perform a Link Sync after a random period between [THR_POWERON_ROUTER_MIN_JITTER_MS, THR_POWERON_ROUTER_MA↩ X_JITTER_MS].

1.2.69 #define THR_POWERON_ROUTER_MAX_JITTER_MS

On power on, during the network start with NVM, a router will perform a Link Sync after a random period between [THR_POWERON_ROUTER_MIN_JITTER_MS, THR_POWERON_ROUTER_MA↩ X_JITTER_MS].

1.2.70 #define THR_POWERON_ED_MAX_JITTER_MS

On power on, during the network start with NVM, an end device will perform a Child Update after a random period between [gThrPowerOnRouterMaxJitterMs, gThrPowerOnRouterMaxJitterMs+gThrPower↩ OnEDMaxJitterMs].

1.2.71 #define THR_PARENT_ROUTE_TO_LEADER_TIMEOUT_MS

The number of seconds a Child waits prior to reattaching in the event its Parent advertises an infinite cost to the Leader.

1.2.72 #define THR_CHILD_ED_KEEP_ALIVE_INTERVAL_MIN_MS

The default periodic interval for REED or End DeviceRxOn to send ChildUpdateRequest.

These values should be less than THR_CHILD_ED_TIMEOUT_PERIOD_SEC

1.2.73 #define THR_CHILD_ED_KEEP_ALIVE_INTERVAL_MAX_MS

The default periodic interval for REED or End DeviceRxOn to send ChildUpdateRequest.

These values should be less than THR_CHILD_ED_TIMEOUT_PERIOD_SEC

1.2.74 #define THR_CONTEXT_REUSE_DELAY_SEC

Network Data Context ID reuse delay.

1.2.75 #define THR_DISCOVERY_EXT_ADDR

The extended address used for discovery.

1.2.76 #define THR_DISCOVERY_KEY

The key used for discovery.

1.2.77 #define THR_DISCOVERY_FRAME_COUNTER

The frame counter used for discovery.

1.2.78 #define THR_DISCOVERY_TIME

Time an originator of a Discovery Request should wait for incoming Discovery Responses on a channel.

1.2.79 #define THR_DISCOVERY_MAX_JITTER

Maximum jitter time used to delay generation of Discovery Responses.

Chapter 2
Thread Network Interface

2.1 Overview

Files

- file thread_network.h

Data Structures

- struct thrDeviceConfig_t

Macros

- #define THR_NWKCAP_CAN_CREATE_NEW_NETWORK
- #define THR_NWKCAP_CAN_BECOME_ACTIVE_ROUTER
- #define THR_NWKCAP_IS_POLLING_END_DEVICE
- #define THR_NWKCAP_IS_FULL_THREAD_DEVICE
- #define THR_NWKCAP_BIT_MASK

Enumerations

- enum thrEvCodesNwkScan_t { gThrEv_NwkScanCnf_Results_c }
- enum thrEvCodesCreate_t {
  gThrEv_NwkCreateCnf_Success_c,
  gThrEv_NwkCreateCnf_Failed_c,
  gThrEv_NwkCreateInd_SelectBestChannel_c,
  gThrEv_NwkCreateInd_GeneratePSKc_c }
- enum thrEvCodesJoin_t {
  gThrEv_NwkJoinInd_Attaching_c,
  gThrEv_NwkJoinCnf_Success_c,
  gThrEv_NwkJoinCnf_Failed_c }
- enum thrEvCodesJoinSelectParent_t {
  gThrEv_NwkSelectParentsInd_ScanStarted_c,
  gThrEv_NwkSelectParentsInd_RcvBeacon_c,
  gThrEv_NwkSelectParentsInd_ScanEnded_c }
- enum thrEvCodesGeneral_t {

gThrEv_GeneralInd_Disconnected_c,
gThrEv_GeneralInd_Connected_c,
gThrEv_GeneralInd_ResetToFactoryDefault_c,
gThrEv_GeneralInd_InstanceRestoreStarted_c,
gThrEv_GeneralInd_RouterSynced_c,
gThrEv_GeneralInd_EndDeviceSynced_c,
gThrEv_GeneralInd_ConnectingStarted_c,
gThrEv_GeneralInd_ConnectingFailed_c,
gThrEv_GeneralInd_ConnectingDeffered_c,
gThrEv_GeneralInd_DeviceIsLeader_c,
gThrEv_GeneralInd_DeviceIsRouter_c,
gThrEv_GeneralInd_DevIsREED_c,
gThrEv_GeneralInd_DevIsFED_c,
gThrEv_GeneralInd_DevIsSED_c,
gThrEv_GeneralInd_RequestGlobalAddr_c,
gThrEv_GeneralInd_GlobalAddrAssigned_c,
gThrEv_GeneralInd_RequestRouterId_c,
gThrEv_GeneralInd_RouterIdAssigned_c,
gThrEv_GeneralInd_RouterIdAssignedFailed_c,
gThrEv_GeneralInd_AllowDeviceToSleep_c,
gThrEv_GeneralInd_DisallowDeviceToSleep_c,
gThrEv_GeneralInd_ChildIdAssigned_c,
gThrEv_GeneralInd_DevIsMED_c,
gThrEv_GeneralInd_ResetMcuTimeout_c }

Functions

- void THR_Task (osaTaskParam_t argument)
- void THR_Init (void)
- thrStatus_t THR_InitAttributes (instanceId_t thrInstId, stackConfig_t *pStackCfg)
- thrStatus_t THR_StartInstance (instanceId_t thrInstId, stackConfig_t *pStackCfg)
- thrStatus_t THR_SetDeviceConfig (instanceId_t thrInstId, thrDeviceConfig_t *pThrDeviceConfig)
- thrStatus_t THR_SetDeviceRole (instanceId_t thrInstID, thrDeviceRole_t thrDeviceRole)
- thrStatus_t THR_NwkScanWithBeacon (instanceId_t thrInstId, thrNwkScan_t *pThrNwkScan)
- thrStatus_t THR_NwkDiscoveryReq (instanceId_t thrInstId, thrNwkDiscoveryReqTxOpt_t *pDiscReqTxOpt, thrDiscoveryRespCb_t pfDiscoveryRespCb)
- thrStatus_t THR_NwkDiscoveryStop (instanceId_t thrInstId)
- thrStatus_t THR_SearchThreadNwkWithAnnounce (instanceId_t thrInstId, uint32_t scanChannelMask, thrAnnounceCb_t pfAnnounceCb)
- thrStatus_t THR_SetBorderRouterIf (instanceId_t thrInstId, ipIfUniqueId_t brIfId)
- thrStatus_t THR_NwkCreate (instanceId_t thrInstId)
- thrStatus_t THR_NwkAttach (instanceId_t thrInstId)
- thrStatus_t THR_NwkJoin (instanceId_t thrInstId, thrJoinDiscoveryMethod_t discMethod)
- thrStatus_t THR_NwkDetach (instanceId_t thrInstId)
- thrStatus_t THR_SoftwareReset (instanceId_t thrInstID, bool_t factoryReset)
- void THR_FactoryReset (void)
- void THR_TimeoutResetMcu (uint32_t timeoutMs, bool_t resetToFactory)
- thrNeighbor_t * THR_GetParent (instanceId_t thrInstID)
- thrNeighbor_t * THR_GetNeighborTable (uint32_t iCount)

- uint16_t **THR_NeighborGetShortByExtAddr** (uint64_t *pEui)
- thrNeighbor_t * THR_NeighborGetByShort (uint16_t shortAddr)
- thrRouterIdSet_t * THR_GetRouterIdSet (instanceId_t thrInstId)
- thrStatus_t THR_LeaderRemoveRouterID (instanceId_t thrInstID, uint32_t routerID)
- thrStatus_t THR_RouterLinkSync (instanceId_t thrInstID, bool_t bOnReset)
- thrStatus_t THR_ChildUpdateToParent (instanceId_t thrInstID)
- thrStatus_t THR_SolicitGlobalAddress (instanceId_t thrInstID)
- thrStatus_t THR_BrPrefixAttrAddEntry (instanceId_t thrInstID, thrOtaBrPrefixSet_t *pEntry)
- thrStatus_t THR_ServiceAttrAddEntry (instanceId_t thrInstID, thrLocalServiceSet_t *pEntry)
- thrStatus_t THR_BrPrefixAttrRemoveEntry (instanceId_t thrInstID, uint8_t prefixLength, uint8_t *pPrefixValue)
- thrStatus_t THR_BrServiceAttrRemoveEntry (instanceId_t thrInstID, uint8_t *pServiceData, uint8_t serviceDataLen, uint8_t *pServerData, uint8_t serverDataLen)
- void THR_BrPrefixAttrGetTable (instanceId_t thrInstID, uint8_t startIndex, uint8_t reqNoOf↩ Elements, uint8_t *pRspNoOfElements, uint8_t *pOutData)
- thrStatus_t THR_BrPrefixAttrRemoveAll (instanceId_t thrInstID)
- thrStatus_t THR_BrPrefixAttrSync (instanceId_t thrInstID)
- thrStatus_t THR_SendProactiveAddressNotification (instanceId_t thrInstId, ipAddr_t *pDestIp↩ Addr)
- uint64_t THR_GenerateExtendedAddress (bool_t privacyAddr)
- thrLqCacheEntry_t * THR_GetRlocToEidMapByEntry (uint32_t entry)

Variables

- thrDeviceConfig_t gaThrDeviceConfig [ ]

2.2 Data Structure Documentation

2.2.1 struct thrDeviceConfig_t

thread device configuration

Data Fields

bool_t	outOfBand↩ Configured	If TRUE than the device is out-of-band configured. • On network creation, it is not used. • On joining, if it is set TRUE the THR_NwkJoin() will perform only the attaching procedure; otherwise it will perform the joining with Commissioner procedure (mesh-cop joining).

uint8_t	outOfBand↩ Channel	Network creation channel. If different form 0, On network creation (THR_NwkCreate()), will OVERRIDE the SCAN channel and only use this channel. Range: 0, 11-26
uint16_t	panId	• On network creation (THR_NwkCreate()), the configured value will be used or if it is set to 0xffff then the device will generate a random pan ID .
uint32_t	scanChannels	The channel mask used for scanning for networks and to discover network parameters (panId, channel, xpan, network name)
uint8_t	xPanId[8]	• On network creation (THR_NwkCreate()), the configured value will be used or if all bytes are 0xff then the device will generate a random extended pan ID. • On joining using out-of-band configuration (outOfBand↩ Configured = TRUE), if all bytes are 0xff then the device won't filter after the extended pan ID; otherwise it uses this extended pan id for filtering.
uint8_t	masterKey[16]	• On network creation (THR_NwkCreate()), the configured value will be used or if all bytes are 0xff then the device will generate a random master key. • On joining using out-of-band configuration (outOfBand↩ Configured = TRUE), the device uses the configured key for communication.
thrOctet16_t	nwkName	On joining with the out-of-band configuration (outOfBand↩ Configured = TRUE), if (outOfBandChannel == 0) and nwk↩ Name.length != 0, the device will filter after network name to find the pan id and channel.
thrPrefixAttr↩ _t	MLprefix	• On network creation (THR_NwkCreate()), the configured value will be used or if all bytes are 0xff then the device will generate a random mesh local prefix.

2.3 Macro Definition Documentation

2.3.1 #define THR_NWKCAP_CAN_CREATE_NEW_NETWORK

Thread Device Capabilities.

The node can create a new network

2.3.2 #define THR_NWKCAP_CAN_BECOME_ACTIVE_ROUTER

The node can become an active router.

2.3.3 #define THR_NWKCAP_IS_POLLING_END_DEVICE

The node is a polling end device (sleepy end device)

2.3.4 #define THR_NWKCAP_IS_FULL_THREAD_DEVICE

The node is a full Thread device (FTD)

2.3.5 #define THR_NWKCAP_BIT_MASK

Thread Device Capabilities bit mask.

2.4 Enumeration Type Documentation

2.4.1 enum thrEvCodesNwkScan_t

Network scan events.

Enumerator

gThrEv_NwkScanCnf_Results_c nwk scan confirm - results

2.4.2 enum thrEvCodesCreate_t

Network Create Events.

Enumerator

gThrEv_NwkCreateCnf_Success_c nwk create confirm - success
gThrEv_NwkCreateCnf_Failed_c nwk create confirm - failed
gThrEv_NwkCreateInd_SelectBestChannel_c nwk create indication - select best channel
gThrEv_NwkCreateInd_GeneratePSKc_c nwk create indication - generate PSKc

2.4.3 enum thrEvCodesJoin_t

Network Join Events.

Enumerator

gThrEv_NwkJoinInd_Attaching_c nwk join indication - attaching
gThrEv_NwkJoinCnf_Success_c nwk join confirm - success
gThrEv_NwkJoinCnf_Failed_c nwk join confirm - failed

2.4.4 enum thrEvCodesJoinSelectParent_t

Network Select Parent when joining with Commissioner.

Enumerator

gThrEv_NwkSelectParentsInd_ScanStarted_c network select parent indication - scan started
gThrEv_NwkSelectParentsInd_RcvBeacon_c network select parent indication - received beacon
gThrEv_NwkSelectParentsInd_ScanEnded_c network select parent indication - scan ended

2.4.5 enum thrEvCodesGeneral_t

Network General Events - warning the order of events impacts the THCI event monitor.

Enumerator

gThrEv_GeneralInd_Disconnected_c general event indication - disconnected
gThrEv_GeneralInd_Connected_c general event indication - connected
gThrEv_GeneralInd_ResetToFactoryDefault_c general event indication - device started with factory defaults
gThrEv_GeneralInd_InstanceRestoreStarted_c general event indication - start restore from reset
gThrEv_GeneralInd_RouterSynced_c general event indication - restored from reset with success for router
gThrEv_GeneralInd_EndDeviceSynced_c general event indication - restored from reset with success for end device
gThrEv_GeneralInd_ConnectingStarted_c general event indication - trying to connect to the network
gThrEv_GeneralInd_ConnectingFailed_c general event indication - failed to connect to the network
gThrEv_GeneralInd_ConnectingDeffered_c general event indication - app must initiate connect action
gThrEv_GeneralInd_DeviceIsLeader_c general event indication - device has leader role

gThrEv_GeneralInd_DeviceIsRouter_c general event indication - device has router role
gThrEv_GeneralInd_DevIsREED_c general event indication - device has REED role
gThrEv_GeneralInd_DevIsFED_c general event indication - device has RX on when idle end device role
gThrEv_GeneralInd_DevIsSED_c general event indication - device has sleepy end device role
gThrEv_GeneralInd_RequestGlobalAddr_c general event indication - request global address
gThrEv_GeneralInd_GlobalAddrAssigned_c general event indication - global address assigned
gThrEv_GeneralInd_RequestRouterId_c general event indication - request router short address
gThrEv_GeneralInd_RouterIdAssigned_c general event indication - router short address assigned
gThrEv_GeneralInd_RouterIdAssignedFailed_c general event indication - failed to received router short address
gThrEv_GeneralInd_AllowDeviceToSleep_c general event indication - allow device to sleep
gThrEv_GeneralInd_DisallowDeviceToSleep_c general event indication - disallow device to sleep
gThrEv_GeneralInd_ChildIdAssigned_c general event indication - child short address assigned
gThrEv_GeneralInd_DevIsMED_c general event indication - device has minimal end device role
gThrEv_GeneralInd_ResetMcuTimeout_c general event indication - reset mcu timeout

2.5 Function Documentation

2.5.1 void THR_Task (osaTaskParam_t *argument*)

Thread application task.

Parameters

in	*argument*	Task private data

Returns

NONE

2.5.2 void THR_Init (void)

Initialize Thread module.

Returns

NONE

2.5.3 thrStatus_t THR_InitAttributes (instanceId_t *thrInstId,* stackConfig_t * *pStackCfg*)

Function that initializes with factory defaults or restores from NVM the Thread Attributes.

Parameters

in	*thrInstId*	Thread instance Id
in	*pStackCfg*	Pointer to stack configuration

Returns

thrStatus_t Result of the operation

2.5.4 thrStatus_t THR_StartInstance (instanceId_t *thrInstId,* stackConfig_t * *pStackCfg*)

Function that starts the Thread instance.

Parameters

in	*thrInstID*	Thread instance ID
in	*pStackCfg*	Pointer to stack configuration

Returns

thrStatus_t Result of the operation

2.5.5 thrStatus_t THR_SetDeviceConfig (instanceId_t *thrInstId,* thrDeviceConfig_t * *pThrDeviceConfig*)

This function is used to set device configuration. This function overwrites the default settings (see app↩ _thread_config.h) with a minimum set of attributes needed to start a node. The application may not call this function if it wants to use the default settings.

Parameters

in	*thrInstID*	Thread instance Id
in	*pThrDevice↩ Config*	Pointer to device configuration

Returns

thrStatus_t Status

2.5.6 thrStatus_t THR_SetDeviceRole (instanceId_t *thrInstID,* thrDeviceRole_t *thrDeviceRole*)

This is a public function used to set the network capabilities for a Thread device.

Parameters

in	*thrInstID*	Thread instance ID
in	*thrDeviceRole*	Thread desired device role.

Returns

thrStatus_t Status

2.5.7 thrStatus_t THR_NwkScanWithBeacon (instanceId_t *thrInstId,* thrNwkScan_t * *pThrNwkScan*)

This function is used to start a network scan using beacon messages. A callback function must be registered (using EVM_RegisterStatic() function) with the gThrEvSet_NwkScan_c set to receive the scan results (see thrNwkScanResults_t message).

Parameters

in	*thrInstID*	Thread instance Id
in	*pThrNwkScan*	Network scan parameters

Returns

thrStatus_t Status

2.5.8 thrStatus_t THR_NwkDiscoveryReq (instanceId_t *thrInstId,* thrNwkDiscoveryReqTxOpt_t * *pDiscReqTxOpt,* thrDiscoveryRespCb_t *pfDiscoveryRespCb*)

This function starts the Thread Discovery Procedure. A callback function must be registered (pf↩ DiscoveryRespCb) to receive the Discovery Responses.

Parameters

in	*thrInstID*	Thread instance Id
in	*pDiscReqTx↩ Opt*	Pointer to Discovery Request Tx options
in	*pfDiscovery↩ RespCb*	Pointer to a callback to receive the Discovery Responses

Returns

thrStatus_t Status

2.5.9 thrStatus_t THR_NwkDiscoveryStop (instanceId_t *thrInstId*)

This function stops the discovery process before timing out.

Parameters

in	*thrInstID*	Thread instance Id

Returns

thrStatus_t Status

2.5.10 thrStatus_t THR_SearchThreadNwkWithAnnounce (instanceId_t *thrInstId,* uint32_t *scanChannelMask,* thrAnnounceCb_t *pfAnnounceCb*)

The device has the Nwk key and searches the thread network using the announcement messages. Only Thread networks that have the same Nwk key will respond. This function could be used to discover the channel and panId of a Thread network, so that to start the attachment process (to perform the out-of-band joining procedure).

Parameters

in	*thrInstID*	Thread instance Id
in	*scanChannel↩ Mask*	Channel mask to scan
in	*pfAnnounceCb*	Pointer to a callback that handles the received announcement messages.

Returns

thrStatus_t Status

2.5.11 thrStatus_t THR_SetBorderRouterIf (instanceId_t *thrInstID,* ipIfUniqueId_t *brIfId*)

This function is used for configuring a device as a Thread Border Router by providing the external interface to use for routing outbound packets

Parameters

in	*thrInstID*	Thread instance Id.
in	*brIfId*	Border router external interface Unique ID

Returns

thrStatus_t status

2.5.12 thrStatus_t THR_NwkCreate (instanceId_t *thrInstId*)

This function is used to create a Thread network (start the device as a leader node). If the PAN ID and channel attributes are configured (panid != 0xFFFF and channel != 0), the function will start the leader using these attributes. Otherwise, the scanChannelMask attribute is used to select the best channel and panID. A callback function is registered (see thread_app_callbacks.h and app_thread_callbacks.c) with the gThrEvSet_NwkCreate_c event set to receive the network creation events(see APP_NwkCreateCb callback). Note that THR_NWKCAP_CAN_CREATE_NEW_NETWORK bit must be set in the network capabilities.

Parameters

in	*thrInstID*	Thread instance Id

Returns

thrStatus_t Status

2.5.13 thrStatus_t THR_NwkAttach (instanceId_t *thrInstId*)

This function is used to perform the attachment procedure using the configured attributes: channel, panId, network key. NOTE:

- It can be used for out-of-band joining.
- A callback function must be registered (using EVM_RegisterStatic() function) with the gThrEv↩ Set_NwkJoin_c event set to receive the network join events.

Parameters

in	*thrInstID*	Thread instance Id

Returns

thrStatus_t Status

2.5.14 thrStatus_t THR_NwkJoin (instanceId_t *thrInstId,* thrJoinDiscovery↩ Method_t *discMethod*)

This function is used to join a device to a thread network and is using all the above functionality. Depending on attributes configuration, it can perform the following actions:

1. join as an end node using commissioning (calling MESHCOP_NwkJoinWithCommissioning()). In this case, the device is NOT out-of-band configured;this means the devIsCommissioned attribute shall be set to FALSE. Depending on discovery method parameter (discMethod), the device will search the panId and channel using Beacon or Thread Discovery messages.

2. attach to a thread network (calling THR_NwkAttach()). In this case, the device must be out-of-band configured;this means the devIsCommissioned attribute shall be set to TRUE. Depending on discovery method parameter (discMethod), the device will search the panId and channel using Beacon or Announcement messages (see THR_SearchThreadNwkWithAnnounce()). A callback function must be registered (using EVM_RegisterStatic() function) with the gThrEvSet_NwkJoin_c event set to receive the network join events.

Parameters

in	*thrInstID*	Thread instance Id
in	*discMethod*	The discovery method (see thrJoinDiscoveryMethod_t)

Returns

thrStatus_t Status

2.5.15 thrStatus_t THR_NwkDetach (instanceId_t *thrInstId*)

This function is used to detach a joined device. The device will keep the network settings but will disconnect itself from the network.

Parameters

in	*thrInstID*	Thread instance Id

Returns

thrStatus_t Status

2.5.16 thrStatus_t THR_SoftwareReset (instanceId_t *thrInstID,* bool_t *factoryReset*)

This function is used to do a thread software reset.

Parameters

in	*thrInstID*	Thread instance Id
in	*factoryReset*	If TRUE, the device will be reseted to factory

Returns

thrStatus_t Status

2.5.17 void THR_FactoryReset (void)

This function is used to reset device to factory default settings.

Returns

NONE

2.5.18 void THR_TimeoutResetMcu (uint32_t *timeoutMs,* bool_t *resetToFactory*)

This function is used to reset the device after a specific timeout.

Parameters

in	*timeoutMs*	Time expressed in milliseconds units. [in] resetToFactory If TRUE, the device will be reseted to factory

Returns

NONE

2.5.19 thrNeighbor_t * THR_GetParent (instanceId_t *thrInstID*)

This is a public function used to get information about parent when node is an end device and REED or about the attaching parent in case of a router. For a leader it will return NULL.

Parameters

in	*thrInstID*	Thread instance ID

Returns

thrNeighbor_t* Pointer to neighbor structure.

2.5.20 thrNeighbor_t * THR_GetNeighborTable (uint32_t *iCount*)

This function is used to get the neighbor from a given index from the neighbor table.

Parameters

in	*iCount*	Entry in neighbor table

Returns

thrNeighbor_t * Pointer to neighbor structure.

2.5.21 thrNeighbor_t * THR_NeighborGetByShort (uint16_t *shortAddr*)

This function is used to get a neighbor's information by its short address.

Parameters

in	*shortAddr*	Neighbor's short address

Returns

thrNeighbor_t Neighbor info

2.5.22 thrRouterIdSet_t * THR_GetRouterIdSet (instanceId_t *thrInstId*)

This function is used to get the thread router ID set.

Parameters

in	*thrInstId*	Thread instance id

Returns

thrRouterIdSet_t* Pointer to router ID set structure

2.5.23 thrStatus_t THR_LeaderRemoveRouterID (instanceId_t *thrInstID,* uint32_t *routerID*)

This function is used to remove a router from network. It should be called only on the leader node.

Parameters

in	*thrInstID*	Thread instance ID
in	*routerID*	The ID of the router to be removed

Returns

thrStatus_t Status

2.5.24 thrStatus_t THR_RouterLinkSync (instanceId_t *thrInstID,* bool_t *bOnReset*)

This function will perform the "link synchronization after reset" or "initial link synchronization" procedure. Must be called only on a router node.

Parameters

in	*thrInstID*	Thread instance Id
in	*bOnReset*	Specify if the router should do the "Router synchronization after reset" procedure or the "Initial Router synchronization" procedure.

Returns

thrStatus_t Status

2.5.25 thrStatus_t THR_ChildUpdateToParent (instanceId_t *thrInstID*)

This function is used to send a ChildUpdate.Request to synchronize the parent with the updated attributes (timeout period, node mode flags TLV).

Parameters

in	*thrInstID*	Thread instance Id

Returns

thrStatus_t Status

2.5.26 thrStatus_t THR_SolicitGlobalAddress (instanceId_t *thrInstID*)

This function solicits a global address from a DHCPv6 server.

Parameters

in	*thrInstID*	Thread instance Id

Returns

thrStatus_t Status

2.5.27 thrStatus_t THR_BrPrefixAttrAddEntry (instanceId_t *thrInstID,* thrOtaBrPrefixSet_t * *pEntry*)

Add a Border router prefix attribute entry.

Parameters

in	*thrInstID*	Thread instance Id
in	*pEntry*	Border router nwk data entry

Returns

thrStatus_t Status

2.5.28 thrStatus_t THR_ServiceAttrAddEntry (instanceId_t *thrInstID,* thrLocalServiceSet_t * *pEntry*)

Add a Border router service attribute entry.

Parameters

in	*thrInstID*	Thread instance Id
in	*pEntry*	Border router service set entry

Returns

thrStatus_t Status

2.5.29 thrStatus_t THR_BrPrefixAttrRemoveEntry (instanceId_t *thrInstID,* uint8_t *prefixLength,* uint8_t * *pPrefixValue*)

Remove Border Router prefix attribute entry.

Parameters

in	*thrInstID*	Thread instance Id
in	*prefixLength*	Prefix size
in	*pPrefixValue*	Prefix value

Returns

thrStatus_t Status

2.5.30 thrStatus_t THR_BrServiceAttrRemoveEntry (instanceId_t *thrInstID,* uint8_t * *pServiceData,* uint8_t *serviceDataLen,* uint8_t * *pServerData,* uint8_t *serverDataLen*)

Remove Service attribute entry.

Parameters

in	*thrInstID*	Thread instance Id
in	*pServiceData*	Pointer to service data
in	*serviceData↩ Len*	Service data size
in	*pServerData*	Pointer to server data
in	*serverDataLen*	Server data size

Returns

thrStatus_t Status

2.5.31 void THR_BrPrefixAttrGetTable (instanceId_t *thrInstID,* uint8_t *startIndex,* uint8_t *reqNoOfElements,* uint8_t * *pRspNoOfElements,* uint8_t * *pOutData*)

Get the BR table, from startIndex to a maximum of reqNoOfElements.

Parameters

in	*thrInstID*	Thread instance Id
in	*startIndex*	Start to search from this index
in	*reqNoOf↩ Elements*	Try to retrieve this many elements
out	*pRspNoOf↩ Elements*	Actual number of elements retrieved
out	*pOutData*	Data buffer to store the information

2.5.32 thrStatus_t THR_BrPrefixAttrRemoveAll (instanceId_t *thrInstID*)

Remove all Border router prefix attribute entries.

Parameters

in	*thrInstID*	Thread instance Id

Returns

thrStatus_t Status

2.5.33 thrStatus_t THR_BrPrefixAttrSync (instanceId_t *thrInstID*)

This function is used to synchronize the Border Router prefixes with the over-the-air network data information. After adding or removing more BR prefixes, this function shall be called to propagate the global network data.

Parameters

in	*thrInstID*	Thread instance Id

Returns

thrStatus_t Status

2.5.34 thrStatus_t THR_SendProactiveAddressNotification (instanceId_t *thrInstId,* ipAddr_t * *pDestIpAddr*)

This function is used to send ADDR_NTF.ntf - Proactive Address Notification. This is useful if the device has changed short address and knows there are devices that likely maintain an address cache of that short address.

Parameters

in	*thrInstId*	Thread instance ID
in	*pDestIpAddr*	Destination address: unicast or multicast

Returns

thrStatus_t Status

2.5.35 uint64_t THR_GenerateExtendedAddress (bool_t *privacyAddr*)

This function generates a random extended mac address

Parameters

in	*privacyAddr*	TRUE if the address should be a privacy address

Return values

uint64_t	extended mac address

2.5.36 thrLqCacheEntry_t * THR_GetRlocToEidMapByEntry (uint32_t *entry*)

This function is used to get a specific entry from eid to rloc mapping cache table.

Parameters

in	*entry*	Entry table number.

Returns

thrLqCacheEntry_t* Pointer to value of the entry, NULL if entry number is not valid

2.6 Variable Documentation

2.6.1 thrDeviceConfig_t gaThrDeviceConfig[]

Thread device configuration.

Chapter 3
Thread Attributes Interface

3.1 Overview

Files

- file thread_attributes.h

Data Structures

- struct thrAttr_t
- struct thrStringAttr_t
- struct thrActiveAttr_t
- struct thrPendingAttr_t
- struct thrOtaBrPrefixSet_t
- struct thrLocalServiceSet_t

Macros

- #define **THR_BR_PREFIX_FLAGS_P_PREFERENCE_MASK**
- #define **THR_BR_PREFIX_FLAGS_P_PREFERRED_MASK**
- #define **THR_BR_PREFIX_FLAGS_P_SLAAC_MASK**
- #define **THR_BR_PREFIX_FLAGS_P_DHCP_MASK**
- #define **THR_BR_PREFIX_FLAGS_P_CONFIGURE_MASK**
- #define **THR_BR_PREFIX_FLAGS_P_DEFAULT_MASK**
- #define **THR_BR_PREFIX_FLAGS_P_ON_MESH_MASK**
- #define **THR_BR_PREFIX_FLAGS_P_ND_DNS_MASK**
- #define **THR_BR_PREFIX_FLAGS_P_PREFERENCE_OFFSET**
- #define **THR_BR_PREFIX_FLAGS_P_PREFERRED_OFFSET**
- #define **THR_BR_PREFIX_FLAGS_P_SLAAC_OFFSET**
- #define **THR_BR_PREFIX_FLAGS_P_DHCP_OFFSET**
- #define **THR_BR_PREFIX_FLAGS_P_CONFIGURE_OFFSET**
- #define **THR_BR_PREFIX_FLAGS_P_DEFAULT_OFFSET**
- #define **THR_BR_PREFIX_FLAGS_P_ON_MESH_OFFSET**
- #define **THR_BR_PREFIX_FLAGS_P_ND_DNS_OFFSET**
- #define **THR_BR_EXT_ROUTE_FLAGS_R_PREF_MASK**
- #define **THR_BR_EXT_ROUTE_FLAGS_R_PREF_OFFSET**
- #define **THR_BR_FLAGS_SET**(flag, value, mask, offset)
- #define **THR_BR_FLAGS_GET**(value, mask, offset)
- #define **THR_FLAGS_PREFERENCE_MEDIUM**
- #define **THR_FLAGS_PREFERENCE_HIGH**
- #define **THR_FLAGS_PREFERENCE_RESERVED**
- #define **THR_FLAGS_PREFERENCE_LOW**
- #define **gNwkAttrId_Undefined_c**
- #define **THR_GetAttr_Channel**(thrInstId)
- #define **THR_GetAttr_PanId**(thrInstId)
- #define **THR_GetAttr_ShortAddr**(thrInstId)

- #define **THR_GetAttr_IsDevConnected**(thrInstId)
- #define **THR_GetAttr_PartionId**(thrInstId)
- #define **THR_GetAttr_NwkCapabilities**(thrInstId)
- #define **THR_GetAttr_IsDevCommissioned**(thrInstId)
- #define **THR_GetAttr_DeviceRole**(thrInstId)
- #define **THR_GetAttr_DeviceType**(thrInstId)
- #define **THR_GetAttr_CommissionerMode**(thrInstId)
- #define **THR_GetAttr_SelBestChEDThreshold**(thrInstId)
- #define **THR_GetAttr_JoinLqiThreshold**(thrInstId)
- #define **THR_GetAttr_PermitJoin**(thrInstId)
- #define **THR_GetAttr_BrDefaultRoute**(thrInstId)
- #define **THR_GetAttr_BrExternalIfPrefix**(thrInstId)
- #define **THR_GetAttr_BrGlobalOnMeshPrefix**(thrInstId)
- #define **THR_GetAttr_ScanChannelMask**(thrInstId)
- #define **THR_GetAttr_SedFastPollInterval**(thrInstId)
- #define **THR_GetAttr_SedPollInterval**(thrInstId)
- #define **THR_GetAttr_SedFastPoll**(thrInstId)
- #define **THR_GetAttr_DoNotGeneratePartId**(thrInstId)
- #define **THR_GetAttr_NwkKeySeq**(thrInstId)
- #define **THR_GetAttr_NvmRestore**(thrInstId)
- #define **THR_GetAttr_NvmAutostart**(thrInstId)
- #define **THR_GetAttr_ChildAddrMask**(thrInstId)
- #define **THR_GetAttr_ScanDuration**(thrInstId)
- #define **THR_GetAttr_DiscReqMacTxOptions**(thrInstId)
- #define **THR_GetAttr_RandExtAddr**(thrInstId)
- #define **THR_GetAttr_ShaHashAddr**(thrInstId)
- #define **THR_GetAttr_RxOnWhenIdle**(thrInstId)
- #define **THR_GetAttr_Weighting**(thrInstId)
- #define **THR_GetAttr_ChildReqFullNwkData**(thrInstId)
- #define **THR_GetAttr_ParentHoldTime**(thrInstId)
- #define **THR_GetAttr_KeySwitchGuardTime**(thrInstId)
- #define **THR_GetAttr_JoinerUdpPort**(thrInstId)
- #define **THR_GetAttr_MinDelayTimer**(thrInstId)
- #define **THR_GetAttr_CommissionerUpdPort**(thrInstId)
- #define **THR_GetAttr_SedTimeoutPeriod**(thrInstId)
- #define **THR_GetAttr_EdTimeoutPeriod**(thrInstId)
- #define **THR_GetAttr_SlaacPolicy**(thrInstId)

Enumerations

- enum thrAttrId_t {
  gNwkAttrId_RandExtAddr_c,
  gNwkAttrId_ShortAddr_c,
  gNwkAttrId_ScanChannelMask_c,
  gNwkAttrId_ScanDuration_c,
  gNwkAttrId_Channel_c,
  gNwkAttrId_PanId_c,
  gNwkAttrId_ExtendedPanId_c,
  gNwkAttrId_PermitJoin_c,
  gNwkAttrId_RxOnIdle_c,
  gNwkAttrId_SEDPollInterval_c,
  gNwkAttrId_UniqueExtAddr_c,
  gNwkAttrId_VendorName_c,
  gNwkAttrId_ModelName_c,
  gNwkAttrId_SWVersion_c,
  gNwkAttrId_StackVersion_c,
  gNwkAttrId_ThreadNwkCapabilites_c,
  gNwkAttrId_NwkName_c,
  gNwkAttrId_DeviceType_c,
  gNwkAttrId_IsDevConnected_c,
  gNwkAttrId_IsDevCommissioned_c,
  gNwkAttrId_PartitionId_c,
  gNwkAttrId_DeviceRole_c,
  gNwkAttrId_NwkMasterKey_c,
  gNwkAttrId_NwkKeySeq_c,
  gNwkAttrId_PSKc_c,
  gNwkAttrId_PSKd_c,
  gNwkAttrId_VendorData_c,
  gNwkAttrId_EDTimeoutPeriod_c,
  gNwkAttrId_MLPrefix_c,
  gNwkAttrId_WhiteListEntry_c,
  gNwkAttrId_KeyRotationInterval_c,
  gNwkAttrId_ChildAddrMask_c,
  gNwkAttrId_SEDTimeoutPeriod_c,
  gNwkAttrId_ChildEDReqFullNwkData_c,
  gNwkAttrId_IsFastPollEnabled_c,
  gNwkAttrId_SEDFastPollInterval_c,
  gNwkAttrId_JoinLqiThreshold_c,
  gNwkAttrId_ProvisioningURL_c,
  gNwkAttrId_SelectBestChEDThreshold_c,
  gNwkAttrId_SteeringData_c,
  gNwkAttrId_KeySwitchGuardTime_c,
  gNwkAttrId_ParentHoldTime_c,
  gNwkAttrId_SecurityPolicy_c,
  gNwkAttrId_NvmRestoreAutoStart_c,
  gNwkAttrId_NvmRestore_c,
  gNwkAttrId_GenericIsaacPolicy_c,
  gNwkAttrId_IeeeAddr_c,

gNwkAttr_MinDelayTime }

Functions

- thrStatus_t THR_InitAttr (instanceId_t thrInstId, void *pDefaultAttr, void *pDefaultStrAttr, void *pDefaultActiveDataSetAttr)
- thrStatus_t THR_GetAttr (instanceId_t thrInstID, thrAttrId_t attrID, uint32_t index, uint32_t *pSize, void *pAttrValue)
- thrStatus_t THR_SetAttr (instanceId_t thrInstID, thrAttrId_t attrID, uint32_t index, uint32_t size, void *pAttrValue)

Variables

- thrAttr_t * gpaThrAttr [ ]
- thrStringAttr_t * gpaThrStringAttr [ ]
- thrActiveAttr_t * gpaThrActiveAttr [ ]
- thrPendingAttr_t * gpaThrPendingAttr [ ]
- ipAddr_t gaServerDataPrefixTbl [ ]
- uint8_t **gaServerDataPrefixLenTbl** [ ]
- borderRouterSet_t **gaThrServerDataBrSetTbl** [ ]
- externalRouteSet_t **gaServerDataExtRouteTbl** [ ]
- thrLocalServiceSet_t * **gpaLocalServiceSetTbl** [ ]
- const uint8_t gLocalServiceSetTblSize

3.2 Data Structure Documentation

3.2.1 struct thrAttr_t

Thread network information base (Thread Nibs) structure:

- 802.15.4 attributes
- thread specific attributes

Data Fields

uint64_t	ieeeAddr	The MAC extended address, also called IEEE address, long address or 64-bit MAC address.
uint64_t	shaHashAddr	MAC address used for commissioning generated from SHA256 hash on the ieeeAddr.
uint64_t	randExtAddr	The random MAC extended address, used for communication inside the network after commissioning.
uint8_t	scanDuration	The scan duration time. This is an exponential scale, as seen in the 802.15.4 specification. Range: 0 - 14. Values greater than 14 will be set to 14, as described in Thread Specification chapter 8.10.2.↩ 10. The user can take into account that the total scanning time also depends on the number of channels scanned. Ex.: for a scanmask of 16 channels, maximum value of scanDuration is 8, meaning 3.75 seconds/channel.
uint16_t	shortAddr	The short address.
bool_t	permitJoin	Permit Join(Router devices only). True = Device is allowing the child to join the network, False = Device is not allowing any child to join the network
bool_t	rxOnWhenIdle	The receiver is ON when the device is in idle state. Set RxOn↩ WhenIdle TRUE for mains-powered (non-sleeping) end-devices. Set this FALSE for sleeping end-devices. When FALSE, end-devices will poll their parent for messages. See sedPollInterval for the polling timeout.
uint32_t	sedPollInterval	The poll interval in milliseconds. This attribute is used only for sleepy end devices
uint32_t	sedFastPoll↩ Interval	The fast poll interval in milliseconds. This attribute is used only for sleepy end devices during the joining procedure
bool_t	isFastPoll↩ Enabled	Specify if the fast polling is enabled.
bool_t	uniqueExtAddr	If is set to TRUE, the device is automatically generated a random extended address.
bool_t	devIs↩ Connected	Specifies if the device is connected or not.
bool_t	devIs↩ Commissioned	If TRUE than the device is commissioned. • On joining calling THR_NwkJoin() with devIs↩ Commissioned == TRUE, the device will perform the attaching procedure using the commissioning settings. Note that a network active scan is performed before attaching. • On joining calling THR_NwkJoin() with devIs↩ Commissioned == FALSE, the device will perform the joining with Commissioner procedure (mesh-cop joining).

uint32_t	sedTimeout↩ Period	The Timeout period used by the parent to consider a sleepy end device (SED) disconnected.
uint32_t	edTimeout↩ Period	The Timeout period used by the parent to consider an end device (ED) disconnected.
bool_t	childEDReq↩ FullNwkData	If it is set TRUE, the end device is requesting the full network data(stable and temporary nwk data). If it is set FALSE, the end device is requesting only the unstable (temporary) network data.
thrDevice↩ Type_t	deviceType	The device type: 0x00 -EndNode, 0x01 - Combo Node.
thrInternal↩ DeviceRole_t	devRole	The device role: 0x01 - SED, 0x02 - MED, 0x03 - FED,0x04 - REED, 0x05 - Router, 0x06 - Leader.
uint8_t	thrNwk↩ Capabilities↩ BitMap	A bitmap that specify network capabilities of this device.
uint32_t	nwkKeySeq	The current network key sequence number.
uint32_t	childAddr↩ Mask[THR_↩ MAX_CHIL↩ D_IDS]	The child address mask.
uint32_t	partitionId	The current partition identifier.
uint8_t	weighting	Leader weight.
bool_t	doNot↩ Generate↩ PartitionId	Avoid random generation of partition ID.
uint8_t	joinLqi↩ Threshold	The joining LQI threshold used to select a potential parent.
uint8_t	selBest↩ ChannelED↩ Threshold	The energy channel threshold to select the best channel when more channels are scan (energy detect scan) to form the network. Note that this is used only if the scanChannelMask attribute includes more than one channel.
uint16_t	joinerUdpPort	Joiner UDP port.
uint16_t	commissioner↩ UdpPort	Commissioner UDP port.
uint32_t	keySwitch↩ GuardTime	The thread Key switch guard time to prevent inadvertent key switching.
uint16_t	parentHold↩ Time	Thread hold time in seconds used by the parent to hold the packets for SED devices.
uint8_t	slaacPolicy	Used SLAAC policy (see thrSlaacPolicy_t)
bool_t	nvmRestore↩ AutoStart	Stack starts automatically with NVM restore after reset.

bool_t	nvmRestore	Restore from NVM.
thrPrefixAttr_t	brGlobalOnMeshPrefix	Global /64 on-Mesh Prefix on Border Router.
bool_t	brDefaultRoute	Default Route of the /64 on-mesh prefix.
thrPrefixAttr_t	brExternalIfPrefix	Global /64 external interface prefix.
uint8_t	discoveryReqMacTxOptions	The default discovery request Mac Tx options (see thrDiscReqTxOptions_t)
uint32_t	minDelayTimer	The minimum accepted time before a Pending Dataset can be installed[s].

3.2.2 struct thrStringAttr_t

Thread network information base (Thread Nibs) structure:

- thread specific string attributes

Data Fields

thrOctet32_t	vendorName	Vendor name.
thrOctet16_t	modelName	Model Name.
thrOctet16_t	swVersion	Software version.
thrOctet64_t	provisioningURL	Provisioning URL.
thrOctet16_t	steeringData	Steering data.
thrOctet32_t	pskD	The passphrase used in authentication procedure - a new Joiner device is the correct one.
uint8_t	stackVersion[6]	Stack version - ReadOnly.
thrOctet64_t	vendorData	Vendor data.
thrOctet64_t	commissionerId	Commissioner ID.

3.2.3 struct thrActiveAttr_t

Data Fields

uint8_t	channel	The current channel.
uint32_t	scanChannelMask	The channels mask used when a network scanning is performed (energy scan, active scan or both); 0x07FFF800 means all 16 channels are used (from 11 to 26).

uint16_t	panId	The PAN identifier (ID). • On network creation (calling THR_NwkCreate()), if it is set 0xffff then the device will generate a random pan ID.
uint8_t	xPanId[8]	The extended PAN ID. • On network creation (calling THR_NwkCreate()), if all bytes are 0xff's then the device will generate a random extended pan ID. • On joining using out-of-band configuration (calling THR_NwkJoin() when devIsCommissioned = TRUE), if xPanId != all ff's the device is using this extended pan id to find the pan ID and channel; otherwise no filter is apply
thrPrefixAttr_t	MLprefix	
uint8_t	nwkMasterKey[16]	The network master key.
thrOctet16_t	nwkName	Network Name. • On joining calling THR_NwkJoin() with devIsCommissioned == TRUE, if nwkName.length != 0, the device will filter after network name to find the network parameters.
uint8_t	pskC[16]	The Pre-Shared Key (PSKc) derived from Commissioning Credential (network password)
uint8_t	securityPolicy	O and N bits without rotation time.
uint32_t	nwkKeyRotationInterval	The current key rotation interval in minutes.
uint8_t	timestamp[8]	First 6 bytes: seconds, last 2 bytes: ticks.

3.2.4 struct thrPendingAttr_t

Data Fields

uint16_t	channel	Pending channel.
uint32_t	scanChannelMask	Pending Channel Mask.

uint16_t	panId	Pending Pan ID.
uint8_t	xPanId[8]	Pending Extended PanId.
thrPrefixAttr_t	mlPrefix	Pending MeshLocal prefix.
uint8_t	nwkMasterKey[16]	Pending Master Key.
thrOctet16_t	nwkName	Pending Network Name.
uint8_t	pskC[16]	Pending PSKc.
uint8_t	securityPolicy	Pending Security Policy bits.
uint32_t	nwkKeyRotationInterval	Pending Key Rotation Interval[sec].
uint8_t	activeTimestamp[8]	Pending Active Timestamp.
uint8_t	timestamp[8]	Pending Timestamp.
uint32_t	delayTimer	Pending Delay Timer[msec].

3.2.5 struct thrOtaBrPrefixSet_t

border router network data attributes - ota format

Data Fields

uint8_t	thrBrPrefixLength	Prefix length in bits.
uint8_t	thrBrPrefixValue[16]	Prefix value.
uint8_t	thrBrPrefixFlags[2]	Flags data - Border Router TLV of the Thread Network Data: byte0 - reserved, byte1 - border router flags (see above BR prefix flags)
uint8_t	thrBrPrefixLifetime[4]	Prefix Data Lifetime (seconds)
bool_t	thrBrPrefixAdvertised	Flag that indicates whether a Border Router TLV SHALL be advertised for prefix in the Network Data.
uint8_t	thrBrExtRouteFlags	Flags data - Has Route TLV of the Thread Network Data (see above BR external route flags)
uint8_t	thrBrExtRouteLifetime[4]	External Route Data Lifetime (seconds)

bool_t	thrBrExt↩ Route↩ Advertised	Flag that indicates whether a Has Route TLV SHALL be advertised for prefix in the Network Data.

3.2.6 struct thrLocalServiceSet_t

Data Fields

uint8_t	thr↩ Senterprise↩ Number[4]	Senterprise number.
uint8_t	thrSservice↩ DataLen	Size of the Sservice data.
uint8_t	thrSservice↩ Data[THR_S↩ ERVICE_DA↩ TA_MAX_L↩ EN]	Sservice data eg "dnsserver".
uint8_t	thrSserver16↩ Addr[2]	Sserver address.
uint8_t	thrSserver↩ DataLen	Size of sserver data (16)
uint8_t	thrSserver↩ Data[THR_S↩ ERVER_DA↩ TA_MAX_L↩ EN]	Sserver data.
uint8_t	thrSserviceId	Sservice id.
bool_t	thrSstable	Sstable mode.

3.3 Enumeration Type Documentation

3.3.1 enum thrAttrId_t

Thread network information base (Thread Nibs) Ids:

- 802.15.4 attributes
- thread specific attributes

Enumerator

gNwkAttrId_RandExtAddr_c Random extended address.
gNwkAttrId_ShortAddr_c Short address.
gNwkAttrId_ScanChannelMask_c Scan channel mask.
gNwkAttrId_ScanDuration_c Scan duration.

gNwkAttrId_Channel_c Channel.
gNwkAttrId_PanId_c Pan id.
gNwkAttrId_ExtendedPanId_c Extended pan id.
gNwkAttrId_PermitJoin_c Permit join.
gNwkAttrId_RxOnIdle_c Rx on when idle.
gNwkAttrId_SEDPollInterval_c Poll interval.
gNwkAttrId_UniqueExtAddr_c Unique extended address (test only)
gNwkAttrId_VendorName_c Vendor name.
gNwkAttrId_ModelName_c Model name.
gNwkAttrId_SWVersion_c Software version.
gNwkAttrId_StackVersion_c Stack version.
gNwkAttrId_ThreadNwkCapabilites_c Nwk capabilities.
gNwkAttrId_NwkName_c Nwk name.
gNwkAttrId_DeviceType_c Device type.
gNwkAttrId_IsDevConnected_c Is device connected.
gNwkAttrId_IsDevCommissioned_c Is device out-of band preconfigured.
gNwkAttrId_PartitionId_c Partition id of the network.
gNwkAttrId_DeviceRole_c Device role.
gNwkAttrId_NwkMasterKey_c Nwk master key.
gNwkAttrId_NwkKeySeq_c Nwk key sequence.
gNwkAttrId_PSKc_c Network credential.
gNwkAttrId_PSKd_c Device password.
gNwkAttrId_VendorData_c Vendor data.
gNwkAttrId_EDTimeoutPeriod_c The timeout period included in the Child ID Request sent to the parent.
gNwkAttrId_MLPrefix_c Mesh local prefix.
gNwkAttrId_WhiteListEntry_c White list entry.
gNwkAttrId_KeyRotationInterval_c Key rotation interval.
gNwkAttrId_ChildAddrMask_c Child address mask.
gNwkAttrId_SEDTimeoutPeriod_c The timeout period included in the Child ID Request sent to the parent.
gNwkAttrId_ChildEDReqFullNwkData_c If it is set TRUE The child End device should request the Full network data.
gNwkAttrId_IsFastPollEnabled_c Is fast poll enabled.
gNwkAttrId_SEDFastPollInterval_c Fast poll interval.
gNwkAttrId_JoinLqiThreshold_c Join lqi threshold.
gNwkAttrId_ProvisioningURL_c A URL for the Joiner to communicate to the user which Commissioning application is best to use to properly provision it to the appropriate service.
gNwkAttrId_SelectBestChEDThreshold_c The energy channel threshold to select the best channel when more channels are scan to form the network.
gNwkAttrId_SteeringData_c Steering data.
gNwkAttrId_KeySwitchGuardTime_c The thread Key switch guard time to prevent inadvertent key switching.
gNwkAttrId_ParentHoldTime_c The hold time period in seconds used by the parent to hold the packets for SED devices.

gNwkAttrId_SecurityPolicy_c O and N bits without the rotation time.
gNwkAttrId_NvmRestoreAutoStart_c Stack starts automatically with NVM restore after reset.
gNwkAttrId_NvmRestore_c Restore from NVM.
gNwkAttrId_SlaacPolicy_c Specifies the policy for generating the IID of an address configured using SLAAC.
gNwkAttrId_IeeeAddr_c MAC IEE Extended address used for SHA256 address generation.
gNwkAttrId_LeaderWeight_c Leader Weight, an 8-bit unsigned integer.
gNwkAttrId_HashIeeeAddr_c SHA256 generated MAC address used during commissioning phase.

gNwkAttrId_DoNotGeneratePartitionId_c Avoid random generation of partition ID.
gNwkAttrId_BrGlobalOnMeshPrefix_c Global /64 on-Mesh Prefix on Border Router.
gNwkAttrId_BrDefaultRouteOnMeshPrefix_c Default Route of the /64 on-mesh prefix.
gNwkAttrId_BrExternalIfPrefix_c Global /64 external interface prefix.
gNwkAttrId_MeshCop_ActiveTimestamp_c Active timestamp.
gNwkAttrId_MeshCop_PendingChannel_c Pending channel.
gNwkAttrId_MeshCop_PendingChannelMask_c Pending Channel Mask.
gNwkAttrId_MeshCop_PendingXpanId_c Pending Extended PanId.
gNwkAttrId_MeshCop_PendingMLprefix_c Pending MeshLocal prefix.
gNwkAttrId_MeshCop_PendingNwkMasterKey_c Pending Master Key.
gNwkAttrId_MeshCop_PendingNwkName_c Pending Network Name.
gNwkAttrId_MeshCop_PendingPanId_c Pending Pan ID.
gNwkAttrId_MeshCop_PendingPSK_c Pending PSKc.
gNwkAttrId_MeshCop_PendingSecurityPolicy Pending Security Policy bits.
gNwkAttrId_MeshCop_PendingNwkKeyRotationInterval_c Pending Key Rotation Interval[sec].
gNwkAttrId_MeshCop_PendingDelayTimer_c Pending Delay Timer[msec].
gNwkAttrId_MeshCop_PendingActiveTimestamp_c Pending Active Timestamp.
gNwkAttrId_MeshCop_PendingTimestamp_c Pending Timestamp.
gNwkAttrId_MeshCop_CommissionerId_c Commissioner string.
gNwkAttr_JoinerUdpPort_c Joiner UDP port.
gNwkAttr_CommissionerUdpPort_c Commissioner UDP port.
gNwkAttr_DiscoveryReqMacTxOptions_c The default discovery request Mac Tx options (see thrDiscReqTxOptions_t)
gNwkAttr_MinDelayTime The minimum accepted time before a Pending Dataset can be installed[s].

3.4 Function Documentation

3.4.1 thrStatus_t THR_InitAttr (instanceId_t *thrInstId,* void * *pDefaultAttr,* void * *pDefaultStrAttr,* void * *pDefaultActiveDataSetAttr*)

Initialize the attributes.

Parameters

in	*thrInstID*	Thread instance Id
in	*pDefaultAttr*	Pointer to the default attributes
in	*pDefaultStrAttr*	Pointer to the default string attributes
in	*pDefault↩ ActiveData↩ SetAttr*	Pointer to the default active data set attributes

Returns

thrStatus_t Thread status - Succes/Fail

3.4.2 thrStatus_t THR_GetAttr (instanceId_t *thrInstID,* thrAttrId_t *attrID,* uint32_t *index,* uint32_t * *pSize,* void * *pAttrValue*)

Get thread attribute value.

Parameters

in	*thrInstID*	Thread instance Id
in	*attrID*	Attribute Id
in	*index*	Index (use zero if it is a scalar attribute)
out	*pSize*	Attribute size
out	*pAttrValue*	Pointer to Attribute Value

Returns

thrStatus_t Thread status - Succes/Fail

3.4.3 thrStatus_t THR_SetAttr (instanceId_t *thrInstID,* thrAttrId_t *attrID,* uint32_t *index,* uint32_t *size,* void * *pAttrValue*)

Set thread attribute value.

Parameters

in	*thrInstID*	Thread instance Id
in	*attrID*	Attribute Id

in	*index*	Index
in	*size*	Attribute size
in	*pAttrValue*	Pointer to Attribute Value

Returns

thrStatus_t Thread status - Succes/Fail

3.5 Variable Documentation

3.5.1 thrAttr_t* gpaThrAttr[]

Thread attributes:

- saved using NVNG

3.5.2 thrStringAttr_t* gpaThrStringAttr[]

Thread string attributes:

- saved using NVNG

3.5.3 thrActiveAttr_t* gpaThrActiveAttr[]

Thread active data set attributes:

- saved using NVNG

3.5.4 thrPendingAttr_t* gpaThrPendingAttr[]

Thread pending data set attributes:

- saved using NVNG

3.5.5 ipAddr_t gaServerDataPrefixTbl[]

Border router network data attributes.

- saved using NVNG

3.5.6 const uint8_t gLocalServiceSetTblSize

Size of the border router service set table.

Chapter 4
Thread Application Callbacks Interface

4.1 Overview

Files

- file thread_app_callbacks.h

Macros

- #define THR_MAX_NWK_JOINING_ENTRIES

Typedefs

- typedef void(∗ registerServiceServerAddr_t) (ipAddr_t ∗pIpAddr)

Functions

- void APP_JoinerSelectNwkWithBeaconCb (void ∗pParam)
- bool_t APP_OutOfBandSelectNwkWithBeaconCb (instanceId_t thrInstId, thrBeaconInfo_t ∗pThr↩ Beacon)
- bool_t APP_MeshcopValidateJoinerAddrCb (instanceId_t thrInstId, ipAddr_t ∗pIpAddr)
- bool_t APP_MeshCopValidateJoinFinCb (instanceId_t thrInstId, meshCopJoinFinTlvs_t ∗pJoin↩ FinTlvs)
- bool_t APP_MeshCopValidateCommissionerCb (instanceId_t thrInstId, meshcopCommIdTlv_↩ t ∗pCommIdTlv)
- bool_t APP_AddressAssignSlaacCb (instanceId_t thrInstId, ipAddr_t ∗pPrefix)
- bool_t **APP_ServiceServerDataCb** (serviceSet_t serviceSet, bool_t bAddService)
- void APP_CriticalExitCb (uint32_t location, uint32_t param)
- bool_t APP_DiscoveryReqCb (instanceId_t thrInstId, uint16_t tlvsSize, uint8_t ∗pTlvs)
- void APP_JoinerDiscoveryRespCb (instanceId_t thrInstId, thrDiscoveryEvent_t event, uint8_t lqi, thrDiscoveryRespInfo_t ∗pDiscoveryRespInfo, meshcopDiscoveryRespTlvs_t ∗pDiscoveryResp↩ Tlvs)
- void APP_JoinerSelectNwkWithAnnounceCb (instanceId_t thrInstId, thrAnnounceEvent_t event, uint8_t lqi, uint16_t tlvsSize, uint8_t ∗pTlvs)
- void APP_GenerateMLPrefixCb (instanceId_t thrInstID, thrPrefixAttr_t ∗pMLprefix)
- void APP_EnableDHCP6Cb (void)
- void APP_BeaconFillCb (instanceId_t thrInstID)

4.2 Macro Definition Documentation

4.2.1 #define THR_MAX_NWK_JOINING_ENTRIES

Maximum number of networks to perform the joining procedure.

4.3 Typedef Documentation

4.3.1 typedef void(∗ registerServiceServerAddr_t) (ipAddr_t ∗pIpAddr)

< Structure for holding pointer to the IPv6 address of a server service advertised in network data

4.4 Function Documentation

4.4.1 void APP_JoinerSelectNwkWithBeaconCb (void ∗ *pParam*)

This function is used to handle the network events during the meshcop joining. This is the callback function used to select the potential network to join when THR_NwkJoin() is called.

Parameters

in	*pParam*	- pointer to event messages (thrEvmParams_t ∗)

4.4.2 bool_t APP_OutOfBandSelectNwkWithBeaconCb (instanceId_t *thrInstId,* thrBeaconInfo_t ∗ *pThrBeacon*)

This is the callback function used to select a thread network (find the panId, channel etc.) when the device in out-of-band configured and THR_NwkJoin(thrInst, gUseMACBeacon_c) is called. This function should filter the received beacons and select a thread network to start the attachment process.

Parameters

in	*thrInstId*	The thread instance ID
in	*pThrBeacon*	Pointer to received Beacon

Returns

TRUE A network has been selected
FALSE No network has been selected

4.4.3 bool_t APP_MeshcopValidateJoinerAddrCb (instanceId_t *thrInstId,* ipAddr_t ∗ *pIpAddr*)

This is the callback function used to check if a Joiner will be accepted by our DTLS server.

Parameters

in	*thrInstId*	The thread instance ID
in	*pIpAddr*	Pointer to client IP address

Returns

TRUE The Joiner is known
FALSE The Joiner is unknown

4.4.4 bool_t APP_MeshCopValidateJoinFinCb (instanceId_t *thrInstId,* meshCopJoinFinTlvs_t * *pJoinFinTlvs*)

Function used to check the TLVs given by the Joiner in the Join Finalization step.

Parameters

in	*thrInstId*	Thread instance ID
in	*pJoinFinTlvs*	Join Finalization TLVs

Returns

TRUE Continue joining
FALSE Otherwise

4.4.5 bool_t APP_MeshCopValidateCommissionerCb (instanceId_t *thrInstId,* meshcopCommIdTlv_t * *pCommIdTlv*)

Function used to check the Commissioner ID. It can be accepted or rejected.

Parameters

in	*thrInstId*	Thread instance ID
in	*pCommIdTlv*	Pointer to the Commissioner ID TLV

Returns

TRUE Allow this Commissioner
FALSE Reject this Commissioner

4.4.6 bool_t APP_AddressAssignSlaacCb (instanceId_t *thrInstId,* ipAddr_t * *pPrefix*)

If slaacPolicy attribute is configured to gThrSlaacManual_c this function serves as a callback to the application to decide if it wants to bind an address with the prefix or not and if so, the application can choose the IID to use with the provided prefix.

Function Documentation

Parameters

in	*thrInstId*	Thread instance ID
	[in/out]	pPrefix Pointer to ip prefix and output to store the IID

Returns

TRUE If the address generated with the prefix and IID should be used by the stack
FALSE If the application does not want to use this prefix and the stack should ignore it

4.4.7 void APP_CriticalExitCb (uint32_t *location,* uint32_t *param*)

If the stack is in a deadlock situation, it calls APP_CriticalExitCb.

Parameters

in	*location*	Address where the Panic occurred
in	*param*	Parameter with extra debug information

4.4.8 APP_DiscoveryReqCb (instanceId_t *thrInstId,* uint16_t *tlvsSize,* uint8_t ∗ *pTlvs*)

This is a callback used by the Application to accept or deny the Discovery Requests. The Discovery Request messages could contain some application specific TLVs, and the APP could have filters based on these TLVs.

Parameters

in	*thrInstId*	Thread instance ID
in	*tlvsSize*	Discovery request TLVs size
in	*pTlvs*	Pointer to Discovery Request TLVs

Returns

TRUE Send the Discovery Response
FALSE Otherwise

4.4.9 void APP_JoinerDiscoveryRespCb (instanceId_t *thrInstId,* thrDiscovery↩ Event_t *event,* uint8_t *lqi,* thrDiscoveryRespInfo_t ∗ *pDiscoveryRespInfo,* meshcopDiscoveryRespTlvs_t ∗ *pDiscoveryRespTlvs*)

This callback can be used by the application to handle and filter the Discovery Response messages. This function is application specific and could build a list of Joiner Routers to start the Meshcop joining process.

Parameters

in	*thrInstId*	Thread instance ID
in	*event*	Discovery event
in	*lqi*	Discovery response packet lqi
in	*pDiscovery↩ RespInfo*	Discovery Response pan information
in	*pDiscovery↩ RespTlvs*	Pointer to Discovery Response TLVs

4.4.10 void APP_JoinerSelectNwkWithAnnounceCb (instanceId_t *thrInstId,* thrAnnounceEvent_t *event,* uint8_t *lqi,* uint16_t *tlvsSize,* uint8_t * *pTlvs*)

This callback handles the announcement messages to select a Thread Network (channel and panId) and start the attachment process.

Parameters

in	*thrInstId*	Thread instance
in	*event*	Announcement events (see thrAnnounceEvent_t)
in	*lqi*	Received packet lqi
in	*tlvsSize*	The size of the received Announce TLVs
in	*pTlvs*	Pointer to Announce TLVs

4.4.11 void APP_GenerateMLPrefixCb (instanceId_t *thrInstID,* thrPrefixAttr_t * *pMLprefix*)

This callback is called by Thread Stack to generate the MLprefix

Parameters

in	*thrInstID*	Thread instance ID
out	*pMLprefix*	ML prefix

4.4.12 void APP_EnableDHCP6Cb (void)

This callback can be used by the application to initialize the callbacks for DHCPv6 module.

4.4.13 void APP_BeaconFillCb (instanceId_t *thrInstID*)

This callback can be used by the application to set the beacon payload.

Parameters

in	*thrInstID*	Thread instance ID

Chapter 5
Thread Types Interface

5.1 Overview

Files

- file thread_types.h

Data Structures

- struct thrOctet16_t
- struct thrOctet32_t
- struct thrOctet64_t
- struct thrPrefixAttr_t
- struct macFilteringNeighborData_t
- struct thrBeaconInfo_t
- struct thrBeaconInfo_t.payload
- struct thrNwkActiveScanEntry_t
- struct thrNwkScan_t
- struct thrNwkScanResults_t
- struct thrNeighbor_t
- struct handleTrackingTable_t
- struct thrIdAssignSet_t
- struct mleOtaTlvLeaderData_t
- struct externalRouteSet_t
- struct borderRouterSet_t
- struct contextIdSet_t
- struct serverTlv_t
- struct serviceSet_t
- struct childVersNbSet_t
- struct serverData_t
- struct nwkDataInterfaceSet_t
- struct thrLqCacheEntry_t
- struct thrAqInterfaceSet_t
- struct thrAddrRegEntry_t
- struct thrChildAddrRegEntry_t
- struct thrChildMcastAddrRegEntry_t
- struct thrLinkSet_t
- struct thrRouteSet_t
- struct thrRouterIdSet_t
- struct thrInterfaceSet_t
- struct thrMacRcvdDiffKeyIndexInd_t
- union thrEventData_t
- struct thrEvmParams_t
- struct thrPskcInputParams_t
- struct thrNwkJoiningEntry_t
- struct thrNwkDiscoveryReqTxOpt_t

Macros

- #define THR_PROTOCOL_VERSION
- #define THREAD_ENTERPRISE_NUMBER
- #define THREAD_ENTERPRISE_NUMBER_ARRAY
- #define THREAD_DNS_SERVICE_TYPE_ID
- #define **THREAD_OTA_SERVICE_TYPE_ID**
- #define THR_MAX_ROUTER_ID
- #define **THR_ROUTER_BITS_SIZE**
- #define **THR_CHILD_BITS_SIZE**
- #define **THR_RSV_BITS_SIZE**
- #define **THR_MAX_ADV_ROUTE_COST**
- #define THR_MAX_POSSIBLE_ROUTERS
- #define THR_ROUTER_MASK_BYTES
- #define THR_MAX_CHILD_IDS
- #define THR_R_ID_ADDR_SHIFT
- #define **THR_R_ID_TO_SHORT_ADDR**(x)
- #define **THR_SHORT_ADDR_TO_R_ID**(x)
- #define **ROUTER_ID_MASK_BYTE**
- #define **ROUTER_ID_MASK**
- #define **CHILD_ID_MASK**
- #define THR_GET_MY_PARENT(chidShortAddr)
- #define **THR_IS_ROUTER**(x)
- #define **THR_IS_END_DEVICE**(x)
- #define THR_IS_MY_CHILD(childShortAddr, parentShortAddr)
- #define **THR_IS_MY_PARENT**(childShortAddr, parentShortAddr)
- #define THR_R_ID_IS_SET_IN_MASK(mask, rId)
- #define THR_NWK_KEY_SIZE
- #define **THR_BEACON_XPAN_DEFAULT_VALUE**
- #define THR_BEACON_J_FLAG_MASK
- #define **THR_BEACON_J_FLAG_OFFSET**
- #define THR_BEACON_N_FLAG_MASK
- #define **THR_BEACON_N_FLAG_OFFSET**
- #define THR_BEACON_VERSION_MASK
- #define **THR_BEACON_VERSION_OFFSET**
- #define THR_BEACON_J_FLAG_GET(byte)
- #define **THR_BEACON_J_FLAG_SET**(byte, flag)
- #define THR_BEACON_N_FLAG_GET(byte)
- #define **THR_BEACON_N_FLAG_SET**(byte, flag)
- #define THR_BEACON_VERSION_GET(byte)
- #define **THR_BEACON_VERSION_SET**(byte, flag)
- #define THR_DISCOVERY_REQ_TLV_J_BIT
- #define THR_DISCOVERY_RESP_TLV_N_BIT
- #define THR_DISC_RSP_VER_SHIFT
- #define **THR_DISC_RSP_VER_SET**(byte, ver)
- #define **THR_DISC_RSP_VER_GET**(byte)
- #define **THR_DISC_RSP_N_SHIFT**
- #define **THR_DISC_RSP_N_SET**(byte, val)
- #define **THR_DISC_RSP_N_GET**(byte)
- #define **THR_DISC_REQ_VER_SHIFT**
- #define **THR_DISC_REQ_VER_SET**(byte, ver)
- #define **THR_DISC_REQ_VER_GET**(byte)
- #define **THR_DISC_REQ_J_SHIFT**
- #define **THR_DISC_REQ_J_SET**(byte, val)
- #define **THR_DISC_REQ_J_GET**(byte)
- #define **THR_SERVICE_DATA_MAX_LEN**

- #define **THR_SERVER_DATA_MAX_LEN**
- #define **THR_MAX_PSKC_LEN**
- #define **THR_PARENT_PRIORITY_OFFSET**
- #define **THR_ML_PREFIX_LEN_BITS**
- #define **gUnusedValue_c**

Typedefs

- typedef uint32_t thrEvCode_t
- typedef void(∗ thrAnnounceCb_t) (instanceId_t thrInstId, thrAnnounceEvent_t event, uint8_t lqi, uint16_t tlvsSize, uint8_t ∗pTlvs)

Enumerations

- enum thrStatus_t {
  gThrStatus_Success_c,
  gThrStatus_Failed_c,
  gThrStatus_InvalidInstance_c,
  gThrStatus_InvalidParam_c,
  gThrStatus_NotPermitted_c,
  gThrStatus_NotStarted_c,
  gThrStatus_NoMem_c,
  gThrStatus_UnsupportedAttr_c,
  gThrStatus_EmptyEntry_c,
  gThrStatus_InvalidValue_c,
  gThrStatus_AlreadyConnected_c,
  gThrStatus_AlreadyCreated_c,
  gThrStatus_NoTimers_c,
  gThrStatus_EntryNotFound_c }
- enum thrInternalDeviceRole_t {
  gThrDevRole_Disconnected,
  gThrDevRole_SED_c,
  gThrDevRole_MED_c,
  gThrDevRole_FED_c,
  gThrDevRole_REED_c,
  gThrDevRole_Router_c,
  gThrDevRole_Leader_c }
- enum thrDeviceRole_t {
  gThrDeviceRole_SED_c,
  gThrDeviceRole_MED_c,
  gThrDeviceRole_FED_c,
  gThrDeviceRole_REED_c }
- enum thrDeviceType_t {
  gThrDevType_EndNode_c,
  gThrDevType_ComboNode_c }
- enum nwkIPAddrType_t {

gLL64Addr_c,
gMLEIDAddr_c,
gRLOCAddr_c,
gGUAAddr_c,
gAnycastAddr_c,
gAnyIpv6_c,
gAllThreadNodes_c }

- enum thrRouterState_t {
  gThrReedIdle_c,
  gThrReedReqRouterId_c,
  gThrReedReatachJitter_c,
  gThrReedReqRouterIdJitter_c,
  gThrRouterIdle_c,
  gThrRouterDownGrdIdJitter_c,
  gThrRouterDownGrd_c }
- enum thrSlaacPolicy_t {
  gThrSlaacRandom_c,
  gThrSlaacManual_c,
  gThrSlaacMlIid_c }
- enum thrCommissionerMode_t {
  gThrCommissionerModeDisabled_c,
  gThrCommissionerModeNative_c,
  gThrCommissionerModeEthernet_c,
  gThrCommissionerModeOnMesh_c,
  gThrCommissionerModeClosing_c }
- enum thrParentPriority_e {
  gThrRouterPriorityMed_c,
  gThrRouterPriorityHigh_c,
  gThrRouterPriorityRsvd_c,
  gThrRouterPriorityLow_c_c }
- enum thrNwkScanType_t {
  gThrNwkScan_EnergyDetect_c,
  gThrNwkScan_ActiveScan_c,
  gThrNwkScan_BothScans_c }
- enum resetCpuStatus_t {
  gResetCpuSuccess_c,
  gResetCpuPending_c }
- enum meshcopSteeringMatch_t {
  gMeshcopSteeringMatchNA_c,
  gMeshcopSteeringMatchNone_c,
  gMeshcopSteeringMatchFfs_c,
  gMeshcopSteeringMatchSpecific_c }
- enum thrEvSets_t {

gThrEvSet_NwkScan_c,
gThrEvSet_NwkCreate_c,
gThrEvSet_NwkJoin_c,
gThrEvSet_NwkSelectParents_c,
gThrEvSet_NwkGeneral_c,
gThrEvSet_NwkCommissioning_c }

- enum thrJoinDiscoveryMethod_t {
  gUseMACBeacon_c,
  gUseThreadDiscovery_c }
- enum thrDiscReqTxOptions_t {
  gThrNoSecurityAtMacLevel_c,
  gThrEncryptedAtMacLevel_c }
- enum thrAnnounceEvent_t {
  gThrSearchThreadNwkStarted_c,
  gThrAnnounceRespRcv_c,
  gThrSearchThreadNwkStopped_c }
- enum thrInstSearchType_t {
  gThrIfUniqueIdSearch_c,
  gThrSlwpInstSearch_c,
  gThrMacInstIdSearch_c,
  gThrInstSearch_c,
  gThrIfHandleSearch_c }

5.2 Data Structure Documentation

5.2.1 struct thrOctet16_t

Specific octet string type, 16 bytes.

Data Fields

uint8_t	length	
uint8_t	aStr[16]	

5.2.2 struct thrOctet32_t

Specific octet string type, 32 bytes.

Data Fields

uint8_t	length	
uint8_t	aStr[32]	

5.2.3 struct thrOctet64_t

Specific octet string type, 64 bytes.

Data Fields

uint8_t	length	
uint8_t	aStr[64]	

5.2.4 struct thrPrefixAttr_t

ML prefix.

Data Fields

ipAddr_t	prefix	
uint8_t	prefixLenBits	

5.2.5 struct macFilteringNeighborData_t

Mac filtering neighbor data.

Data Fields

uint64_t	extended↩ Address	
uint16_t	shortAddress	
uint8_t	linkIndicator	
bool_t	blockNeighbor	

5.2.6 struct thrBeaconInfo_t

Thread Beacon Info.

Data Fields

uint64_t	address	MAC extended address.
uint16_t	panid	PAN ID.
macAbsAddr↩ ModeType_t	addrType	MAC address type: short or long (usually long)
uint8_t	channel	received on channel
uint8_t	lqi	received Lqi
uint8_t	unused	
instanceId_t	slwpInstId	6lowpan instanec ID

uint32_t	payloadSize	beacon payload size
struct thrBeaconInfo_t	payload	

5.2.7 struct thrBeaconInfo_t.payload

Data Fields

uint8_t	protocolId	thread protocol ID
uint8_t	flags	the beacon flags
uint8_t	nwkName[16]	network name
uint8_t	xpanId[8]	extended PAN ID
uint8_t	aTlvs[ ]	where beacon tlv starts

5.2.8 struct thrNwkActiveScanEntry_t

Network Discovery Entry - Each entry represents a Thread network.

Data Fields

uint16_t	numOfRcvdBeacons	number of received beacons on that channel
uint16_t	panid	PAN ID.
uint8_t	channel	received channel
uint8_t	lqi	link quality indicator

5.2.9 struct thrNwkScan_t

This structure is used to perform a network scan.

Data Fields

uint32_t	scanChannelsMask	What channels to scan; 0x07FFF800 means all 16 channels are used (from 11 to 26)
thrNwkScanType_t	scanType	what scan should be performed : energy, active or both
uint8_t	scanDuration	This is an exponential scale, as seen in the 802.15.4 specification (Range:1 - 14)

uint16_t	maxThrNwk↩ ToDisc	

5.2.10 struct thrNwkScanResults_t

The Network scan results.

Data Fields

thrNwkScan_t	scanInfo	
uint8_t	numOf↩ EnergyDetect↩ Entries	
uint8_t *	pEnergy↩ DetectList	One byte for each channel. Only the channels from scanInfo.↩ scanChannelsMask should be handled; the rest of the channels are zeros
uint8_t	numOfNwk↩ ScanEntries	Number of discovered network performing an active scan.
thrNwk↩ ActiveScan↩ Entry_t	nwkScanList[ ]	

5.2.11 struct thrNeighbor_t

Thread Neighbor.

Data Fields

uint64_t	extended↩ Address	Extended Address.
uint32_t	timestamp	Last Time of Communication.
uint32_t	timeoutSec	Device Timeout value.
uint16_t	shortAddress	Short Address.
uint8_t	inLinkMargin	Link Margin of incoming frames from neighbor.
uint8_t	outLinkQuality	Link Quality of sent frames to neighbor.
uint8_t	mode	Device mode.
uint8_t	state	Device state.
uint8_t	txFailure	Number of consecutive transmission failures.
uint8_t	mleReqCount	Number of consecutive MLE Req trans sent.

5.2.12 struct handleTrackingTable_t

Handle Tracking Table Entry.

Data Fields

uint64_t	destAddr	link layer address destination
uint8_t	msduHandle	message handle
macAbsAddr↩ ModeType_t	destAddrMode	link layer address mode

5.2.13 struct thrIdAssignSet_t

Thread ID Assignment set.

Data Fields

uint32_t	thrReuseTime	time interval after which the ID can be reused
uint8_t	thrOwner↩ Eui[gLlayer↩ AddrEui64_c]	link layer address of the ID owner

5.2.14 struct mleOtaTlvLeaderData_t

Leader Data TLV - Over the Air mapping structure.

Data Fields

uint8_t	type	TLV Type.
uint8_t	length	Length of Leader Data TLV.
uint8_t	partitionId[4]	Network Segment Identifier.
uint8_t	weighting	Weighting value for the network fragment.
uint8_t	dataVersion	Version of the Network Data.
uint8_t	stableData↩ Version	Stable Version of the Network Data.
uint8_t	leaderId	Network Leader Router ID.

5.2.15 struct externalRouteSet_t

External Route Set.

Data Fields

uint16_t	brShortAddr	Border Router short address.
uint8_t	hasRouteFlags	Border Router external route flags (Value of R_preference)
uint8_t	brPrefixIndex	Border Route prefix index.
uint8_t	brDomainId	Domain ID.
uint8_t	brLifetime[4]	Entry lifetime.
bool_t	isStable	TRUE - if prefix is valid more than THR_NWK_DATA_MIN_S↩ TABLE_LIFETIME_SEC FALSE - otherwise.
bool_t	bAdvertised	TRUE - Prefix was advertised in the Thread network, FALSE - otherwise.

5.2.16 struct borderRouterSet_t

Border router (BR) Set.

Data Fields

uint16_t	brShortAddr	Border Router short address.
uint8_t	brPrefix↩ Flags[2]	Byte 0: BR Flags; Byte 1: Bits 0-6 Reserved, 7 ND_DNS bit;.
uint8_t	brPrefixIndex	Border Route prefix index.
uint8_t	brDomainId	Domain ID.
uint8_t	brLifetime[4]	Entry lifetime.
bool_t	bIsStable	TRUE - if prefix is valid more than THR_NWK_DATA_MIN_S↩ TABLE_LIFETIME_SEC, FALSE - otherwise.
bool_t	bAdvertised	TRUE - Prefix was advertised in the Thread network, FALSE - otherwise.

5.2.17 struct contextIdSet_t

Context Id Set.

Data Fields

uint8_t	contextFlags	
uint8_t	contextLength	Length of context address.
uint8_t	contextPrefix↩ Index	Prefix index corresponding to context.

bool_t	isStable	TRUE - if prefix is valid more than THR_NWK_DATA_MIN_S↩ TABLE_LIFETIME_SEC, FALSE - otherwise.
uint32_t	removeTstamp	Timestamp after which context can be used only for decompression.

5.2.18 struct serverTlv_t

Data Fields

uint8_t	sServer16[2]	Server's short address.
uint8_t	sDataLen	Length of service data.
uint8_t	sServer↩ Data[THR_S↩ ERVER_DA↩ TA_MAX_L↩ EN]	

5.2.19 struct serviceSet_t

Data Fields

uint8_t	sFlags	Service flags.
uint8_t	sEntNb[4]	Enterprise number.
uint8_t	sDataLen	Service Data length.
uint8_t	sData[THR_↩ SERVICE_D↩ ATA_MAX_↩ LEN]	Service data.
serverTlv_t	sServer	Server TLV.
bool_t	bLocalService	Whether the service is advertised by us.
bool_t	bIsStable	Whether the service is stable.

5.2.20 struct childVersNbSet_t

Child Version Number Set.

Data Fields

uint16_t	childShortAddr	Child short address.
bool_t	childStable↩ Only	TRUE - Child requires only stable data, FALSE - otherwise.
uint8_t	childVersion	Child's version of network data.
uint32_t	childRet↩ Tstamp	Child's retry timestamp.

5.2.21 struct serverData_t

Server Data.

Data Fields

ipAddr_t *	pPrefixTbl	Pointer to Prefix in Prefix Table.
uint8_t *	pPrefixLenTbl	Pointer to Prefix length.
external↩ RouteSet_t *	pExtRouteTbl	Pointer to External Route.
borderRouter↩ Set_t *	pBRSetTbl	Pointer to External Route Set.

5.2.22 struct nwkDataInterfaceSet_t

Thread Network Data Structure.

Data Fields

ipAddr_t *	pPrefixTbl	Pointer to Prefix Table.
uint8_t *	pPrefixLenTbl	Pointer to Prefix Length Table.
childVersNbSet_t *	pChildVersNbSet	Pointer to Children Version Number Set.
borderRouterSet_t *	pBRSetTbl	Pointer to Valid Prefix (Border Router) Set.
externalRouteSet_t *	pExtRouteTbl	Pointer to External Route Set.
contextIdSet_t *	pContextTbl	Pointer to 6LoWPAN Context ID Set.
serviceSet_t *	pServiceSetTbl	Pointer to Service Set.
serverData_t	serverData	Server Data.
mleOtaTlvLeaderData_t	leaderData	Leader Data TLV.

5.2.23 struct thrLqCacheEntry_t

Data Fields

ipAddr_t	eid	IP Address.
uint16_t	address16	Short address.
uint8_t	discoveryTimeout	The time remaining for waiting for responses to an Address Query, or zero if there is no outstanding Address Query.
uint8_t	discoveryFail	The number of consecutive Address Query messages for which no corresponding response was received before discoveryTimeout expires.
uint32_t	retryTimeout	The time a device must wait before sending another Address Query message.
uint32_t	ageSec	Last usage of cache entry.

5.2.24 struct thrAqInterfaceSet_t

Data Fields

ipPktInfo_t *	pIpPktInfo↩ Buffer	Pointer to the outstanding IP packet.
tmrTimerID_t	addrDiscTimer	Timer ID for address discovery.
uint32_t	minClient↩ LastTransa↩ Time	Last transaction time.
uint8_t	mAddrNotify↩ MlEid[8]	Mesh Local EID.

5.2.25 struct thrAddrRegEntry_t

Thread sleepy child ID table entry.

Data Fields

uint8_t	addrIid[8]	Interface identifier.
uint8_t	contextId	Context ID.

5.2.26 struct thrChildAddrRegEntry_t

Thread RFD child address registration table.

Data Fields

uint8_t	neighborIdx	Entry in neighbor table.
thrAddrReg↩ Entry_t	childAddr↩ Entry[THR_↩ CHILD_AD↩ DR_REG_E↩ NTIRES]	Registered IID.

5.2.27 struct thrChildMcastAddrRegEntry_t

Thread RFD child Multicast address registration table.

Data Fields

uint8_t	neighborIdx	Entry in neighbor table.
ipAddr_t	multicastAddr[THR_CHILD_MCAST_ADDR_REG_ENTRIES]	

5.2.28 struct thrLinkSet_t

Thread routing Link set.

Data Fields

uint32_t	thrLinkAge	
uint16_t	thrShortAddr	
uint8_t	thrLinkMargin	
uint8_t	thrOutgoingQual	

5.2.29 struct thrRouteSet_t

Thread routing Route set.

Data Fields

uint16_t	thrMultiHopRouterId	
uint16_t	thrNextHopRouterId	
uint8_t	thrMultihopRouteCost	
uint8_t	thrRouteStatus	

5.2.30 struct thrRouterIdSet_t

Thread routing Router ID set.

Data Fields

uint8_t	thrIdSeqNb	Sequence number.
uint8_t	thrIdSet[THR↩ _ROUTER_↩ MASK_BYT↩ ES]	Router ID Set.

5.2.31 struct thrInterfaceSet_t

Structure with all Thread routing parameters for an interface.

Data Fields

thrRouteSet_t *	pThread↩ RoutingTbl	Pointer to Routing Table.
thrLinkSet_t *	pThreadLink↩ Set	Pointer to link set.
uint32_t	dgradeTstamp	Timestamp for when downgrading to REED.
uint16_t	deviceShort↩ Addr	Device's short address.
tmrTimerID_t	singleShot↩ TmrId	Timer id for single shot operations.
tmrTimerID_t	periodicTmrId	Timer id for periodic operations.
uint8_t	thrRouterCount	Number of routers in network.
bool_t	bIsLeader	TRUE - if device is Leader, FALSE - otherwise.
bool_t	bIsInit	TRUE - if is initialized, FALSE - otherwise.
thrRouter↩ State_t	devState	Device's state.
uint8_t	leaderCost	Route cost to Leader.
thrRouterId↩ Set_t	threadRouter↩ IdSet	Router ID Set.

5.2.32 struct thrMacRcvdDiffKeyIndexInd_t

Mac Key Index.

Data Fields

instanceId_t	macInstId	MAC instance ID.
uint8_t	keyIdMode	Key ID mode.
uint8_t	keyIndex	Key index.

5.2.33 union thrEventData_t

Thread event data.

Data Fields

thrNwkScan↩ Results_t	nwkScanCnf	network scan confirm - result
thrMacRcvd↩ DiffKeyIndex↩ Ind_t	thrMacRcvd↩ DiffKeyIndex↩ Ind	the MAC received a different key index data
thrBeacon↩ Info_t	nwkSelect↩ ParentsInd	network select parents indication

5.2.34 struct thrEvmParams_t

Thread event parameters header.

Data Fields

thrEvCode_t	code	Event Code.
uint32_t	thrInstId	Instance Id.
uint16_t	eventDataSize	Event Data Size.
uint16_t	reserved	Reserved for future use.
thrEventData↩ _t *	pEventData	pointer to event data

5.2.35 struct thrPskcInputParams_t

Structure used to specify input parameters for PSKc generation.

Data Fields

uint8_t *	pPskcStr	PSKc string.
uint32_t	pskcStrLen	PSKc string length.
uint8_t *	pXpanId	Extended PAN ID.
uint8_t *	pNwkName	Network name.
uint32_t	nwkNameLen	Network name length.
uint8_t *	pPskcOut	PSKc.

5.2.36 struct thrNwkJoiningEntry_t

Data Fields

uint8_t	euiAddr[8]	Link layer address.
uint8_t	aXpanId[8]	Extended PAN ID.
uint8_t	channel	Channel.
meshcop↩ Steering↩ Match_t	steeringMatch	Steering Data match.
uint16_t	panId	PAN ID.
uint16_t	joinerUDPPort	if not used, it will be set to 0x0000
uint16_t	commissioner↩ UDPPort	if not used, it will be set to 0x0000

5.2.37 struct thrNwkDiscoveryReqTxOpt_t

Discovery Request TX parameters.

Data Fields

thrDiscReq↩ TxOptions_t	discReqTxOpt	
uint32_t	scanChannel↩ Mask	the scan channel mask (0x07FFF800 means all 16 channels are used).
uint16_t	destPanId	destination PAN ID (it can be 0xFFFF or a specific PAN ID)
uint8_t	flags	flags from Discovery Request TLV: THR_DISCOVERY_REQ_↩ TLV_J_BIT or zero. Note that the Protocol Version will be always added
uint8_t	extraTlvs↩ Length	extra TLV length. More TLVs can be added in the payload (eg extended pan ID, application specific TLVs). Maximum 70 bytes.
uint8_t *	pExtraTlvs	pointer to extra TLV

5.3 Macro Definition Documentation

5.3.1 #define THR_PROTOCOL_VERSION

The Thread protocol version.

5.3.2 #define THREAD_ENTERPRISE_NUMBER

Thread Enterprise number.

5.3.3 #define THREAD_ENTERPRISE_NUMBER_ARRAY

Thread Enterprise number.

5.3.4 #define THREAD_DNS_SERVICE_TYPE_ID

Supported Service Type IDs.

5.3.5 #define THR_MAX_ROUTER_ID

Maximum Router ID.

5.3.6 #define THR_MAX_POSSIBLE_ROUTERS

Maximum number of thread routers.

5.3.7 #define THR_ROUTER_MASK_BYTES

The maximum bytes of the router mask.

5.3.8 #define THR_MAX_CHILD_IDS

The maximum child ID.

5.3.9 #define THR_R_ID_ADDR_SHIFT

Thread Router Id <-> Short address conversion.

5.3.10 #define THR_GET_MY_PARENT(*chidShortAddr*)

Macro for determining the address of a parent based on the child short.
Input: child short address (uin16_t)

5.3.11 #define THR_IS_MY_CHILD(*childShortAddr, parentShortAddr*)

Macro for determining if an node is the devices child.

Input: node short address (uin16_t) parent short address (uint16_t)

5.3.12 #define THR_R_ID_IS_SET_IN_MASK(*mask, rld*)

Check if the router ID is set in the mask.

5.3.13 #define THR_NWK_KEY_SIZE

The Network key size.

5.3.14 #define THR_BEACON_J_FLAG_MASK

Permit join flag mask and offset.

5.3.15 #define THR_BEACON_N_FLAG_MASK

Native commissioner flag mask and offset.

5.3.16 #define THR_BEACON_VERSION_MASK

Beacon Version mask and offset.

5.3.17 #define THR_BEACON_J_FLAG_GET(*byte*)

Thread Beacon Permit Join Flag Macros.

5.3.18 #define THR_BEACON_N_FLAG_GET(*byte*)

Thread Beacon Native Commissioner Flag Macros.

5.3.19 #define THR_BEACON_VERSION_GET(*byte*)

Thread Beacon Permit Join Flag Macros.

5.3.20 #define THR_DISCOVERY_REQ_TLV_J_BIT

Thread Discovery Request/Response TLV bits.

Joiner Flag bit in the byte

5.3.21 #define THR_DISCOVERY_RESP_TLV_N_BIT

Native Commissioner bit in the byte.

5.3.22 #define THR_DISC_RSP_VER_SHIFT

Thread Discovery Request/Response bits and version.

5.4 Typedef Documentation

5.4.1 typedef uint32_t thrEvCode_t

Thread event code.

5.4.2 typedef void(∗ thrAnnounceCb_t) (instanceId_t thrInstId, thrAnnounceEvent_t event, uint8_t lqi, uint16_t tlvsSize, uint8_t ∗pTlvs)

The announcement callback.

5.5 Enumeration Type Documentation

5.5.1 enum thrStatus_t

Thread status.

5.5.2 enum thrInternalDeviceRole_t

Device roles.

Enumerator

gThrDevRole_Disconnected Device is disconnected.
gThrDevRole_SED_c Sleepy End Device, no routing capability.
gThrDevRole_MED_c Minimal End Device, no routing capability.
gThrDevRole_FED_c Full End Device, address discovery and no routing capability.
gThrDevRole_REED_c Router eligible end device (REED)

gThrDevRole_Router_c Router device.
gThrDevRole_Leader_c Leader device.

5.5.3 enum thrDeviceRole_t

Device roles.

Enumerator

gThrDeviceRole_SED_c Sleepy End Device, no routing capability.
gThrDeviceRole_MED_c Minimal End Device, no routing capability.
gThrDeviceRole_FED_c Full End Device, has routing capability.
gThrDeviceRole_REED_c Router eligible end device (REED)

5.5.4 enum thrDeviceType_t

Device types.

Enumerator

gThrDevType_EndNode_c The node can be sleepy or non-sleepy end device (no routing capability)
gThrDevType_ComboNode_c The node can have any device role above.

5.5.5 enum nwkIPAddrType_t

IP Address Types.

Enumerator

gLL64Addr_c Link-Local 64 address (the IID is MAC Extended address Which is not the factory-assigned IEEE EUI-64,)
gMLEIDAddr_c Mesh-Local Endpoint Identifier address (the IID is randmon)
gRLOCAddr_c Routing Locator address (the IID encodes the Router and Child IDs.)
gGUAAddr_c Global Unicast Address.
gAnycastAddr_c Anycast IPv6 addresses.
gAnyIpv6_c All IPv6 address.
gAllThreadNodes_c All Thread nodes address.

5.5.6 enum thrRouterState_t

REED and route states.

5.5.7 enum thrSlaacPolicy_t

The Stateless Address Autoconfiguration (SLAAC) policy.

Enumerator

gThrSlaacRandom_c the addresses is randomly generate
gThrSlaacManual_c it is provided by the application
gThrSlaacMlIid_c use ML-EID address

5.5.8 enum thrCommissionerMode_t

Thread Commissioner mode.

Enumerator

gThrCommissionerModeDisabled_c Commissioner disabled (normal thread node)
gThrCommissionerModeNative_c Native (802.15.4) commissioner.
gThrCommissionerModeEthernet_c Ethernet commissioner.
gThrCommissionerModeOnMesh_c The commissioner is on mesh network. A thread node can become a commissioner at run time
gThrCommissionerModeClosing_c The Commissioner is in closing mode.

5.5.9 enum thrParentPriority_e

parent priority

5.5.10 enum thrNwkScanType_t

Scan type structure.

Enumerator

gThrNwkScan_EnergyDetect_c Energy Detect only.
gThrNwkScan_ActiveScan_c Beacon request only.
gThrNwkScan_BothScans_c Energy detect and beacon request.

5.5.11 enum resetCpuStatus_t

reset CPU status enum

5.5.12 enum meshcopSteeringMatch_t

Enumerator

gMeshcopSteeringMatchNA_c Matching not performed.
gMeshcopSteeringMatchNone_c No matching.
gMeshcopSteeringMatchFfs_c Matched a 0xFF mask.
gMeshcopSteeringMatchSpecific_c Matched specific bits.

5.5.13 enum thrEvSets_t

Thread Event Sets.

Enumerator

gThrEvSet_NwkScan_c network scan event set
gThrEvSet_NwkCreate_c network create event set
gThrEvSet_NwkJoin_c network join event set
gThrEvSet_NwkSelectParents_c network select parent event set
gThrEvSet_NwkGeneral_c network general event set
gThrEvSet_NwkCommissioning_c network commissioning event set

5.5.14 enum thrJoinDiscoveryMethod_t

Thread Discovery method.

Enumerator

gUseMACBeacon_c use MAC beacons for discovery
gUseThreadDiscovery_c use Thread Discovery request

5.5.15 enum thrDiscReqTxOptions_t

Discovery Request TX options.

Enumerator

gThrNoSecurityAtMacLevel_c no security is used at Mac level

gThrEncryptedAtMacLevel_c encrypted with the well-known key and Extended Address at Mac level

5.5.16 enum thrAnnounceEvent_t

The announce events used by the thrAnnounceCb_t callback.

5.5.17 enum thrInstSearchType_t

Thread Instance search type.

Chapter 6
Thread Commissioning Interface

6.1 Overview

Files

- file thread_meshcop_mgmt.h

Data Structures

- struct expectedJoinerEntry_t
- struct meshcopCredentialInput_t
- struct meshCopStateTlv_t
- struct meshCopVendorNameTlv_t
- struct meshCopVendorModelTlv_t
- struct meshCopVendorSwVerTlv_t
- struct meshCopVendorDataTlv_t
- struct meshCopStackVersionTlv_t
- struct meshCopProvUrlTlv_t
- struct meshCopJoinFinTlvs_t
- struct meshCopChannelTlv_t
- struct meshCopChannelMaskTlv_t
- struct meshCopCountTlv_t
- struct meshCopPeriodTlv_t
- struct meshCopEnergyListTlv_t
- struct meshCopScanDurationTlv_t
- struct meshCopDiscoveryReqTlv_t
- struct meshCopDiscoveryRespTlv_t
- struct meshCopDiscoveryTlv_t
- struct meshCopNwkChannelTlv_t
- struct meshCopNwkPanIdTlv_t
- struct meshCopNwkXPanIdTlv_t
- struct meshCopNwkNameTlv_t
- struct meshCopPskcTlv_t
- struct meshCopNwkMasterKeyTlv_t
- struct meshCopNwkKeySeqTlv_t
- struct meshCopNwkMlUlaTlv_t
- struct meshCopSteeringTlv_t
- struct meshCopBrLocTlv_t
- struct meshcopCommIdTlv_t
- struct meshCopCommSessIdTlv_t
- struct meshCopGetTlv_t
- struct meshCopActiveTimestampTlv_t
- struct meshCopCommissionerUdpPortTlv_t
- struct meshCopJoinerUdpPortTlv_t
- struct meshCopPendingTimestampTlv_t
- struct meshCopSecurityPolicyTlv_t
- struct meshCopMacExtendedAddressTlv_t

- struct meshCopDelayTimerTlv_t
- struct meshcopDiscoveryRespTlvs_t
- struct thrDiscoveryRespInfo_t
- struct meshcopHandlers_t
- struct meshcopNwkFormParams_t
- struct meshcopMgmtParams_t

Macros

- #define **MESHCOP_ENABLED**
- #define **MESHCOP_O_MASK**
- #define **MESHCOP_N_MASK**
- #define **MESHCOP_R_MASK**
- #define **MESHCOP_C_MASK**
- #define **MESHCOP_B_MASK**
- #define **MESHCOP_STATE_ACCEPT**
- #define **MESHCOP_STATE_REJECT**
- #define **MESHCOP_STATE_PENDING**
- #define **TLV_TYPE_LEN**
- #define **MESHCOP_MAX_PSK_LEN**
- #define **TLV_NETWORK_PANID_LEN**
- #define **TLV_NETWORK_XPANID_LEN**
- #define **TLV_NETWORK_KEY_LEN**
- #define **TLV_NETWORK_KEY_SEQ_LEN**
- #define **TLV_NETWORK_ML_ULA_LEN**
- #define **MESHCOP_MAX_COMM_ID_LEN**
- #define **MESHCOP_NETWORK_NAME_MAX_LEN**

Typedefs

- typedef void(∗ meshcopDiagnosticHandlerCb_t) (meshcopDiagnosticType_t meshcopDiagType, meshcopDiagnosticDir_t dir, uint8_t ∗pEui, uint8_t ∗pTlvs, uint32_t tlvsLen)
- typedef void(∗ thrDiscoveryRespCb_t) (instanceId_t thrInstId, thrDiscoveryEvent_t event, uint8_t lqi, thrDiscoveryRespInfo_t ∗pDiscoveryRespInfo, meshcopDiscoveryRespTlvs_t ∗pDiscoveryRespTlvs)
- typedef void(∗ meshcopHandlerCb_t) (meshcopHandlers_t ∗pIdHandlerEntry, uint8_t ∗pTlvs, uint32_t tlvsLen)

Enumerations

- enum meshCopTlv_t {
  gMeshCopTlvChannel_c,
  gMeshCopTlvPanID_c,
  gMeshCopTlvXpanID_c,
  gMeshCopTlvNwkName_c,
  gMeshCopTlvPskc_c,
  gMeshCopTlvNwkMasterKey_c,
  gMeshCopTlvNwkKeySeq_c,
  gMeshCopTlvNwkMlUla_c,
  gMeshCopTlvSteeringData_c,
  gMeshCopTlvBorderRouterLoc_c,
  gMeshCopTlvCommID_c,
  gMeshCopTlvCommSessId_c,
  gMeshCopTlvSecPolicy_c,
  gMeshCopTlvGet_c,
  gMeshCopTlvActiveTimestamp_c,
  gMeshCopTlvCommissionerUdpPort_c,
  gMeshCopTlvState_c,
  gMeshCopTlvJoinerDtlsEnc_c,
  gMeshCopTlvJoinerUdpPort_c,
  gMeshCopTlvJoinerAddr_c,
  gMeshCopTlvJoinerRouterLoc_c,
  gMeshCopTlvJoinerRouterKEK_c,
  gMeshCopTlvProvisioningUrl_c,
  gMeshCopTlvVendorName_c,
  gMeshCopTlvVendorModel_c,
  gMeshCopTlvVendorSwVer_c,
  gMeshCopTlvVendorData_c,
  gMeshCopTlvVendorStackVer_c,
  gMeshCopTlvPendingTimestamp_c,
  gMeshCopTlvDelayTimer_c,
  gMeshCopTlvChannelMask_c,
  gMeshCopTlvCount_c,
  gMeshCopTlvPeriod_c,
  gMeshCopTlvScanDuration_c,
  gMeshCopTlvEnergyList_c,
  gMeshCopTlvChannelPages_c,
  gMeshCopTlvDiscoveryReq_c,
  gMeshCopTlvDiscoveryResp_c,
  gMeshCopTlvMacExtAddress_c,
  gMeshCopTlvFuture_c }
- enum meshcopEuiMask_t {

gMeshcopEuiMaskAllZeroes_c,
gMeshcopEuiMaskAllFFs_c,
gMeshcopEuiMaskExpectedJoinerList_c }

- enum thrEvCodesComm_t {
  gThrEv_MeshCop_JoinerDiscoveryStarted_c,
  gThrEv_MeshCop_JoinerDiscoveryFailed_c,
  gThrEv_MeshCop_JoinerDiscoveryFailedFiltered_c,
  gThrEv_MeshCop_JoinerDiscoverySuccess_c,
  gThrEv_MeshCop_JoinerDtlsSessionStarted_c,
  gThrEv_MeshCop_JoinerDtlsError_c,
  gThrEv_MeshCop_JoinerError_c,
  gThrEv_MeshCop_JoinerAccepted_c,
  gThrEv_MeshCop_CommissionerPetitionStarted_c,
  gThrEv_MeshCop_CommissionerPetitionAccepted_c,
  gThrEv_MeshCop_CommissionerPetitionRejected_c,
  gThrEv_MeshCop_CommissionerPetitionError_c,
  gThrEv_MeshCop_CommissionerKeepAliveSent_c,
  gThrEv_MeshCop_CommissionerError_c,
  gThrEv_MeshCop_CommissionerJoinerDtlsSessionStarted_c,
  gThrEv_MeshCop_CommissionerJoinerDtlsError_c,
  gThrEv_MeshCop_CommissionerJoinerAccepted_c,
  gThrEv_MeshCop_CommissionerNwkDataSynced_c,
  gThrEv_MeshCop_CommissionerBrDtlsSessionStarted_c,
  gThrEv_MeshCop_CommissionerBrDtlsError_c,
  gThrEv_MeshCop_CommissionerBrError_c,
  gThrEv_MeshCop_CommissionerBrAccepted_c,
  gThrEv_MeshCop_BrCommissionerDtlsSessionStarted_c,
  gThrEv_MeshCop_BrCommissionerDtlsError_c,
  gThrEv_MeshCop_BrCommissionerAccepted_c,
  gThrEv_MeshCop_BrCommissionerDataRelayedInbound_c,
  gThrEv_MeshCop_BrCommissionerDataRelayedOutbound_c,
  gThrEv_MeshCop_JoinerrouterJoinerDataRelayedInbound_c,
  gThrEv_MeshCop_JoinerrouterJoinerDataRelayedOutbound_c,
  gThrEv_MeshCop_JoinerrouterJoinerAccepted_c,
  gThrEv_MeshCop_StartVendorProvisioning_c }
- enum meshcopDiagnosticDir_t {
  gMeshcopDiagnosticOut_c,
  gMeshcopDiagnosticIn_c }
- enum meshcopDiagnosticType_t {
  gMeshcopDiagnosticJoinFinReq_c,
  gMeshcopDiagnosticJoinFinRsp_c,
  gMeshcopDiagnosticJoinEntReq_c,
  gMeshcopDiagnosticJoinEntRsp_c,
  gMeshcopDiagnosticCloseNotify_c,
  gMeshcopDiagnosticLog_c }

- enum thrDiscoveryEvent_t {
  gThrDiscoveryStarted_c,
  gThrDiscoveryRespRcv_c,
  gThrDiscoveryStopped_c }

Functions

- thrStatus_t **MESHCOP_StartCommissioner** (instanceId_t thrInstId)
- thrStatus_t MESHCOP_StartNativeCommissionerScan (instanceId_t thrInstId)
- bool_t MESHCOP_StopCommissioner (instanceId_t thrInstId, bool_t updateNwk)
- bool_t MESHCOP_AddExpectedJoiner (instanceId_t thrInstId, uint8_t *pEui, uint8_t *pPsk, uint32_t pskLen, bool_t selected)
- expectedJoinerEntry_t * **MESHCOP_GetExpectedJoiner** (instanceId_t thrInstId, uint8_t *pHashEui, uint8_t *pEui)
- bool_t MESHCOP_RemoveExpectedJoiner (instanceId_t thrInstId, uint8_t *pHashEui, uint8_t *pEui)
- void MESHCOP_RemoveAllExpectedJoiners (instanceId_t thrInstId)
- void MESHCOP_SyncSteeringData (instanceId_t thrInstId, meshcopEuiMask_t euiMask)
- meshcopSteeringMatch_t MESHCOP_CheckSteeringData (meshCopSteeringTlv_t *pSteeringDataTlv)
- void MESHCOP_SetCommissionerCredential (instanceId_t thrInstId, meshcopCredentialInput_t *pParams)
- void MESHCOP_SetDiagHandler (instanceId_t thrInstId, meshcopDiagnosticHandlerCb_t pfTlvsHandler)
- uint8_t * MESHCOP_AddTlvs (instanceId_t thrInstanceID, uint8_t *pStart, uint64_t *pMask, bool_t usePending, bool_t noSecPolicy)
- uint32_t MESHCOP_GetTlvsLen (instanceId_t thrInstanceID, uint64_t *pMask, bool_t usePending, bool_t noSecPolicy)
- uint8_t MESHCOP_RegisterBrServerAddr6 (instanceId_t thrInstId, ipIfUniqueId_t ifId, ipAddr_t *pAddr)
- void MESHCOP_NwkJoinWithCommissioning (instanceId_t thrInstId, thrNwkJoiningEntry_t *pNwkJoiningList, uint32_t nbOfNwkJoiningEntries)
- nwkStatus_t MESHCOP_Set (instanceId_t thrInstId, uint8_t *pTlvs, uint32_t tlvsLength, meshcopHandlerCb_t pfSetCb)
- nwkStatus_t MESHCOP_Get (instanceId_t thrInstId, uint8_t *pTlvs, uint32_t tlvsLength, meshcopHandlerCb_t pfGetCb)
- nwkStatus_t **MESHCOP_SendNetworkForm** (meshcopNwkFormParams_t *pNwkFormParams)
- nwkStatus_t MESHCOP_MgmtSendPanIdQuery (instanceId_t thrInstId, uint32_t channelMask, uint16_t panId, meshcopHandlerCb_t pfHandlerCb, ipAddr_t *pIpAddr)
- nwkStatus_t MESHCOP_MgmtSendEdScan (instanceId_t thrInstId, uint32_t channelMask, uint32_t count, uint32_t period, uint32_t scanDuration, meshcopHandlerCb_t pfHandlerCb, ipAddr_t *pIpAddr)
- nwkStatus_t MESHCOP_MgmtSendAnnounceBegin (instanceId_t thrInstId, uint16_t commissionerSessionId, uint32_t channelMask, uint32_t count, uint16_t period, ipAddr_t *pIpAddr)
- nwkStatus_t **MESHCOP_MgmtCommSet** (meshcopMgmtParams_t *pParams)
- nwkStatus_t MESHCOP_MgmtActiveSet (meshcopMgmtParams_t *pParams)
- nwkStatus_t MESHCOP_MgmtPendingSet (meshcopMgmtParams_t *pParams)
- nwkStatus_t MESHCOP_MgmtCommGet (meshcopMgmtParams_t *pParams)
- nwkStatus_t MESHCOP_MgmtActiveGet (meshcopMgmtParams_t *pParams)
- nwkStatus_t MESHCOP_MgmtPendingGet (meshcopMgmtParams_t *pParams)
- nwkStatus_t **MESHCOP_SendUdpRxNtf** (ipAddr_t *pSrcIpAddr, uint16_t pktLength, uint16_t

srcPort, uint16_t dstPort, void *pPayload)
- nwkStatus_t **MESHCOP_SendUdpTxNtf** (ipAddr_t *pDstIpAddr, uint16_t pktLength, uint16_t srcPort, uint16_t dstPort, void *pPayload)

Variables

- list_t gThrExpectedJoinerList
- thrCommissionerMode_t gMeshcopCommissionerMode

6.2 Data Structure Documentation

6.2.1 struct expectedJoinerEntry_t

This entry defines a Joiner.

Data Fields

uint8_t	aHashEui[8]	Extended address of the Joiner(hash)
uint8_t	aPsk[32]	Password of the Joiner.
uint8_t	pskLen	Length in byte of the password.
bool_t	selected	This Joiner is used in computing steering or not.
bool_t	ffEntry	This entry represents all addresses.

6.2.2 struct meshcopCredentialInput_t

Structure used to specify input parameters for PSKc generation on commissioner.

Data Fields

uint8_t *	pPskcStr	Pointer to the human readable password.
uint32_t	pskcStrLen	Size of the human readable password.
uint8_t *	pXpanId	Pointer to the extended pan id.
uint8_t *	pNwkName	Pointer to the network name.
uint32_t	nwkNameLen	Size of the network name buffer.

6.2.3 struct meshCopStateTlv_t

State TLV.

Data Fields

uint8_t	type	Tlv type.

uint8_t	len	Tlv length.
uint8_t	state	State value.

6.2.4 struct meshCopVendorNameTlv_t

Vendor name TLV.

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	vendorName[ ]	Vendor name.

6.2.5 struct meshCopVendorModelTlv_t

Vendor model TLV.

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	vendorModel[ ]	Vendor model.

6.2.6 struct meshCopVendorSwVerTlv_t

Vendor software version TLV.

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	vendorSwVer[ ]	Vendor software version.

6.2.7 struct meshCopVendorDataTlv_t

Vendor data TLV.

Data Fields

uint8_t	type	Tlv type.

uint8_t	len	Tlv length.
uint8_t	vendorData[ ]	Vendor dame.

6.2.8 struct meshCopStackVersionTlv_t

Vendor stack TLV.

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	vendorOui[3]	Organization unique identifier.
uint8_t	majMin	Major and minor version numbers of the Thread Stack.
uint8_t	revBuild[2]	Revision and build numbers of the Thread Stack.

6.2.9 struct meshCopProvUrlTlv_t

Provisioning URL TLV.

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	provUrl[ ]	Provisioning URL.

6.2.10 struct meshCopJoinFinTlvs_t

Joiner Finalization TLVs.

Data Fields

meshCop↩ StateTlv_t *	pState	Pointer to the state tlv.
meshCop↩ VendorName↩ Tlv_t *	pVendorName	Pointer to the vendor name.

meshCop↩ VendorModel↩ Tlv_t *	pVendorModel	Pointer to the vendor model.
meshCop↩ VendorSwVer↩ Tlv_t *	pVendorSwVer	Pointer to the vendor software version.
meshCop↩ VendorData↩ Tlv_t *	pVendorData	Pointer to the vendor data.
meshCop↩ StackVersion↩ Tlv_t *	pVendor↩ StackVer	Pointer to the vendor stack version.
meshCop↩ ProvUrlTlv_t *	pProvUrl	Pointer to the provisioning url.

6.2.11 struct meshCopChannelTlv_t

Channel TLV.

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	channelPage	Channel page.
uint8_t	channel[2]	Channel.

6.2.12 struct meshCopChannelMaskTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	channelPage	Channel page.
uint8_t	maskLength	Channel mask length.

uint8_t	channel↩ Mask[4]	Channel mask.

6.2.13 struct meshCopCountTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	count	Count.

6.2.14 struct meshCopPeriodTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	period[2]	Period between successive scans.

6.2.15 struct meshCopEnergyListTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	aList[ ]	Energy list start.

6.2.16 struct meshCopScanDurationTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	scan↩ Duration[2]	The scan duration in MAC units.

6.2.17 struct meshCopDiscoveryReqTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	verFlags	Version flags.
uint8_t	reserved	Reserved.

6.2.18 struct meshCopDiscoveryRespTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	verFlags	Version flags.
uint8_t	reserved	Reserved.

6.2.19 struct meshCopDiscoveryTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	value[ ]	Start of the discovery tlvs.

6.2.20 struct meshCopNwkChannelTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	channel	Channel.

6.2.21 struct meshCopNwkPanIdTlv_t

Data Fields

uint8_t	type	Tlv type.

uint8_t	len	Tlv length.
uint8_t	panId[TLV_↩ NETWORK_↩ PANID_LEN]	Pan id.

6.2.22 struct meshCopNwkXPanIdTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	xPanId[TLV↩ _NETWORK↩ _XPANID_L↩ EN]	Extended pan id.

6.2.23 struct meshCopNwkNameTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	nwkName[ ]	Network name.

6.2.24 struct meshCopPskcTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	pskc[ ]	Commissioner credential.

6.2.25 struct meshCopNwkMasterKeyTlv_t

Data Fields

uint8_t	type	Tlv type.

uint8_t	len	Tlv length.
uint8_t	masterKey[TLV_NETWORK_KEY_LEN]	Master key.

6.2.26 struct meshCopNwkKeySeqTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	keySeq[TLV_NETWORK_KEY_SEQ_LEN]	Key sequence.

6.2.27 struct meshCopNwkMlUlaTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	mlUla[TLV_NETWORK_ML_ULA_LEN]	Mesh local prefix.

6.2.28 struct meshCopSteeringTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	filter[ ]	Filter bytes.

6.2.29 struct meshCopBrLocTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	addr[2]	Short address in network order.

6.2.30 struct meshcopCommIdTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	commId[ ]	Commissioner id.

6.2.31 struct meshCopCommSessIdTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	id[2]	Commissioner session id in network order.

6.2.32 struct meshCopGetTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	aIds[ ]	List of tlv ids.

6.2.33 struct meshCopActiveTimestampTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	seconds[6]	Seconds part.

uint8_t	ticks[2]	Ticks part.

6.2.34 struct meshCopCommissionerUdpPortTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	aPort[2]	Port number in network order.

6.2.35 struct meshCopJoinerUdpPortTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	aPort[2]	Port number in network order.

6.2.36 struct meshCopPendingTimestampTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	seconds[6]	Seconds part.
uint8_t	ticks[2]	Ticks part.

6.2.37 struct meshCopSecurityPolicyTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	rotation↩ Interval[2]	Key rotation interval in network order.

uint8_t	policy	Policy bits.

6.2.38 struct meshCopMacExtendedAddressTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	aExtended↩ Address[8]	Extended address.

6.2.39 struct meshCopDelayTimerTlv_t

Data Fields

uint8_t	type	Tlv type.
uint8_t	len	Tlv length.
uint8_t	time↩ Remaining[4]	Timer value in netowrk byte order[ms].

6.2.40 struct meshcopDiscoveryRespTlvs_t

Discovery TLVs.

Data Fields

meshCop↩ Discovery↩ RespTlv_t *	pDiscRespTlv	Pointer to discovery response tlv.
meshCop↩ NwkXPanId↩ Tlv_t *	pXpanIdTlv	Pointer to extended pan id tlv.
meshCop↩ NwkName↩ Tlv_t *	pNwkNameTlv	Pointer to network name tlv.

meshCop↩ SteeringTlv_t *	pSteering↩ DataTlv	Pointer to steering data tlv.
meshCop↩ JoinerUdp↩ PortTlv_t *	pJoinerUdp↩ PortTlv	Pointer to joiner udp port tlv.
meshCop↩ Commissioner↩ UdpPortTlv_t *	p↩ Commissioner↩ UdpPortTlv	Pointer to Commissioner udp port tlv.

6.2.41 struct thrDiscoveryRespInfo_t

Discovery Response message.

Data Fields

uint32_t	LQI	LQI of the packet.
uint16_t	panId	Pan id from where the discovery response packet came.
uint8_t	aEui[8]	Extended address of the sender of the discovery response packet.
uint8_t	channel	Channel number from where the discovery packet came.

6.2.42 struct meshcopHandlers_t

Structure defining the MESHCOP handler.

Data Fields

uint32_t	id	Callback Id.
uint32_t	secondId	Optional parameter.
void *	pfCallback	Callback used by the application to receive TLVs.
bool_t	keep	Keep or erase handler after the first execution.
bool_t	used	The status of this entry.

6.2.43 struct meshcopNwkFormParams_t

Structure defining the parameters of MESHCOP_SendNetworkForm.

Data Fields

uint8_t	instanceId	Thread instance Id.
uint8_t	networkNameSize	Network name length.
uint8_t	masterKeySize	Master key length.
uint8_t	pskcSize	PSKC length.
uint8_t *	pNwkName	Pointer to network name.
uint8_t *	pMasterKey	Pointer to master key.
uint8_t *	pPskc	Pointer to PSKC.
meshcopHandlerCb_t	pfGetCb	Pointer to handler function.
uint8_t	channel	Channel.

6.2.44 struct meshcopMgmtParams_t

Structure defining the parameters used for management commands.

Data Fields

instanceId_t	thrInstId	Thread instance Id.
uint8_t *	pTlvs	Pointer to the TLVs to be sent.
uint32_t	tlvsLength	Length of the TLVs buffer.
meshcopHandlerCb_t	pfCb	Pointer to the callback.
ipAddr_t *	pDstIpAddr	Pointer to the IP of the destination.

6.3 Typedef Documentation

6.3.1 typedef void(* meshcopDiagnosticHandlerCb_t) (meshcopDiagnosticType_t meshcopDiagType, meshcopDiagnosticDir_t dir, uint8_t *pEui, uint8_t *pTlvs, uint32_t tlvsLen)

Callback used to send meshcop diagnostics

Parameters

in	*meshcopDiagType*	Diagnostics type
in	*dir*	Direction of packet
in	*pEui*	Pointer to eui address

in	*pTlvs*	Pointer to tlvs
in	*tlvsLen*	Tlvs length

Returns

NONE

6.3.2 typedef void(∗ thrDiscoveryRespCb_t) (instanceId_t thrInstId, thrDiscoveryEvent_t event, uint8_t lqi, thrDiscoveryRespInfo_t ∗pDiscoveryRespInfo, meshcopDiscoveryRespTlvs_t ∗pDiscoveryRespTlvs)

The Discovery Response Callback used by the application to receive the Discovery Responses received during the Discovery process

Parameters

in	*thrInstId*	Thread instance ID
in	*event*	Discovery event type
in	*lqi*	LQI of the Discovery packet received
in	*pDiscovery↩ RespInfo*	Pointer to a structure containing information about the received Discovery response packet
in	*pDiscovery↩ RespTlvs*	Pointer to the Discovery response tlvs

Returns

NONE

6.3.3 typedef void(∗ meshcopHandlerCb_t) (meshcopHandlers_t ∗pIdHandlerEntry, uint8_t ∗pTlvs, uint32_t tlvsLen)

Callback used by the application to receive TLVs

Parameters

in	*pIdHandler↩ Entry*	Pointer to MESHCOP handler

in	*pTlvs*	Pointer to TLVs location
in	*tlvsLen*	TLVs length

Returns

NONE

6.4 Enumeration Type Documentation

6.4.1 enum meshCopTlv_t

TLV types.

6.4.2 enum meshcopEuiMask_t

Bloom filter mode.

Enumerator

gMeshcopEuiMaskAllZeroes_c Don't allow any device.
gMeshcopEuiMaskAllFFs_c Allow all devices.
gMeshcopEuiMaskExpectedJoinerList_c Allow only expected joiners (see expected joiners list)

6.4.3 enum thrEvCodesComm_t

Network Commissioner Events.

Enumerator

gThrEv_MeshCop_JoinerDiscoveryStarted_c Joiner has started discovery.
gThrEv_MeshCop_JoinerDiscoveryFailed_c No Thread networks/routers found.
gThrEv_MeshCop_JoinerDiscoveryFailedFiltered_c Joiner Routers found, but device is filtered.
gThrEv_MeshCop_JoinerDiscoverySuccess_c Network selected.
gThrEv_MeshCop_JoinerDtlsSessionStarted_c Started DTLS session to commissioner (sent Hello)
gThrEv_MeshCop_JoinerDtlsError_c DTLS session error - all DTLS errors, e.g. : incorrect PSKd

gThrEv_MeshCop_JoinerError_c All other non-DTLS errors (e.g. : Joiner Router failed to send credentials)
gThrEv_MeshCop_JoinerAccepted_c Joiner has received credentials.
gThrEv_MeshCop_CommissionerPetitionStarted_c Petitioning has started.
gThrEv_MeshCop_CommissionerPetitionAccepted_c Petition success.
gThrEv_MeshCop_CommissionerPetitionRejected_c Petition rejected.

gThrEv_MeshCop_CommissionerPetitionError_c Other errors in petitioning (did not get PET response)

gThrEv_MeshCop_CommissionerKeepAliveSent_c Generated after each KA.

gThrEv_MeshCop_CommissionerError_c Errors during generating KA or other errors on the commissioner session.

gThrEv_MeshCop_CommissionerJoinerDtlsSessionStarted_c A Joiner sent Hello.

gThrEv_MeshCop_CommissionerJoinerDtlsError_c DTLS session error - all DTLS errors, e.g. : incorrect PSKd

gThrEv_MeshCop_CommissionerJoinerAccepted_c Joiner accepted.

gThrEv_MeshCop_CommissionerNwkDataSynced_c generated after the commissioner changes the Nwk data

gThrEv_MeshCop_CommissionerBrDtlsSessionStarted_c started DTLS session to BR (sent Hello)

gThrEv_MeshCop_CommissionerBrDtlsError_c DTLS session error - all DTLS errors, e.g. : incorrect PSKc

gThrEv_MeshCop_CommissionerBrError_c All Other errors non-DTLS errors when communicating with the BR.

gThrEv_MeshCop_CommissionerBrAccepted_c BR session established.

gThrEv_MeshCop_BrCommissionerDtlsSessionStarted_c Commissioner sent Hello.

gThrEv_MeshCop_BrCommissionerDtlsError_c DTLS session error - all DTLS errors, e.g. : incorrect PSKc

gThrEv_MeshCop_BrCommissionerAccepted_c BR session established.

gThrEv_MeshCop_BrCommissionerDataRelayedInbound_c After each relay from BR to Thread.

gThrEv_MeshCop_BrCommissionerDataRelayedOutbound_c After each relay to BR from Thread.

gThrEv_MeshCop_JoinerrouterJoinerDataRelayedInbound_c After each relay from Joiner to Commissioner.

gThrEv_MeshCop_JoinerrouterJoinerDataRelayedOutbound_c After each relay to Joiner from Commissioner.

gThrEv_MeshCop_JoinerrouterJoinerAccepted_c Before providing the security material to the Joiner.

gThrEv_MeshCop_StartVendorProvisioning_c Device entered Joiner Provisioning mode.

6.4.4 enum meshcopDiagnosticDir_t

Enumerator

gMeshcopDiagnosticOut_c The packet was sent.

gMeshcopDiagnosticIn_c The packet was received.

6.4.5 enum meshcopDiagnosticType_t

Enumerator

gMeshcopDiagnosticJoinFinReq_c JOIN_FIN.req packet.
gMeshcopDiagnosticJoinFinRsp_c JOIN_FIN.rsp packet.
gMeshcopDiagnosticJoinEntReq_c JOIN_ENT.req packet.
gMeshcopDiagnosticJoinEntRsp_c JOIN_ENT.rsp packet.
gMeshcopDiagnosticCloseNotify_c DTLS close notify packet.
gMeshcopDiagnosticLog_c Logging.

6.4.6 enum thrDiscoveryEvent_t

Discovery event received by the Discovery Response Callback.

Enumerator

gThrDiscoveryStarted_c The discovery mechanism has been started.
gThrDiscoveryRespRcv_c Discovery response packet has been received.
gThrDiscoveryStopped_c Discovery mechanism has been completed.

6.5 Function Documentation

6.5.1 thrStatus_t MESHCOP_StartNativeCommissionerScan (instanceId_t *thrInstId*)

This function is used to start the scan process on behalf of the Native Commissioner.

Parameters

in	*thrInstId*	Thread instance ID

Returns

thrStatus_t Status

6.5.2 bool_t MESHCOP_StopCommissioner (instanceId_t *thrInstId,* bool_t *updateNwk*)

This function is used to stop the Commissioner on this device.

Parameters

in	*thrInstId*	Thread instance ID
in	*updateNwk*	Send information into the network

Returns

bool_t TRUE - if the stop operation succeeded FALSE - otherwise

6.5.3 bool_t MESHCOP_AddExpectedJoiner (instanceId_t *thrInstId,* uint8_t * *pEui,* uint8_t * *pPsk,* uint32_t *pskLen,* bool_t *selected*)

Add a Joiner to the expected joiners list.

Parameters

in	*thrInstId*	Thread instance ID
in	*pEui*	Pointer to the extended address of the Joiner
in	*pPsk*	Pointer to given pskc
in	*pskLen*	Length of given pskc
in	*selected*	Use this entry or not

Returns

TRUE The Joiner was scanned successfully
FALSE The Joiner was not scanned successfully

6.5.4 bool_t MESHCOP_RemoveExpectedJoiner (instanceId_t *thrInstId,* uint8_t * *pHashEui,* uint8_t * *pEui*)

Remove an expected joiner from the gThrExpectedJoinerList list.

Parameters

in	*thrInstId*	Thread instance ID
in	*pHashEui*	Pointer to the hash extended address(optional)
in	*pEui*	Pointer to the extended address(optional)

Returns

TRUE Item was found
FALSE Item was not found

6.5.5 void MESHCOP_RemoveAllExpectedJoiners (instanceId_t *thrInstId*)

Remove all expected joiners from the gThrExpectedJoinerList list.

Parameters

in	*thrInstId*	Thread instance ID

6.5.6 void MESHCOP_SyncSteeringData (instanceId_t *thrInstId*, meshcopEuiMask_t *euiMask*)

Sync the steering data on the network with our device.

Parameters

in	*thrInstId*	Thread instance ID
in	*euiMask*	Specify which devices will be steered

Returns

none

6.5.7 meshcopSteeringMatch_t MESHCOP_CheckSteeringData (meshCopSteeringTlv_t * *pSteeringDataTlv*)

Check if this device is in the received steering data.

Parameters

in	*pSteering↩ DataTlv*	Pointer to the Steering Data TLV

Returns

meshcopSteeringMatch_t Matching type

6.5.8 void MESHCOP_SetCommissionerCredential (instanceId_t *thrInstId*, meshcopCredentialInput_t * *pParams*)

Function used to compute and set the PSKc, network name, extended pan Id attributes on the commissioner.

Parameters

in	*thrInstId*	Thread instance id
in	*pParams*	Pointer to the input parameters

6.5.9 void MESHCOP_SetDiagHandler (instanceId_t *thrInstId,* meshcopDiagnosticHandlerCb_t *pfTlvsHandler*)

Function used to set the function which will handle tlvs received during the commissioning process.

Parameters

in	*thrInstId*	Thread instance id
in	*pfTlvsHandler*	Pointer to the tlvs function handler

6.5.10 uint8_t * MESHCOP_AddTlvs (instanceId_t *thrInstanceID,* uint8_t * *pStart,* uint64_t * *pMask,* bool_t *usePending,* bool_t *noSecPolicy*)

Function used add TLV information into buffer.

Parameters

in	*thrInstId*	Thread instance id
in	*pStart*	Pointer to the start of the buffer
in	*pMask*	Pointer to the mask array
in	*usePending*	Boolean variable used to request data from pending set
in	*noSecPolicy*	Internal use: add any TLV w/o taking care of security policy

Returns

uint8_t* Pointer to the buffer after addition

6.5.11 uint32_t MESHCOP_GetTlvsLen (instanceId_t *thrInstanceID,* uint64_t * *pMask,* bool_t *usePending,* bool_t *noSecPolicy*)

Function used to get the length of the TLVs requested in mask.

Parameters

in	*thrInstId*	Thread instance id

in	*pStart*	Pointer to the start of the buffer
in	*pMask*	Pointer to the mask array
in	*usePending*	Boolean variable used to request data from pending set
in	*noSecPolicy*	Do not take care of security policy

Returns

uint32_t Length of the TLVs requested in the mask

6.5.12 uint8_t MESHCOP_RegisterBrServerAddr6 (instanceId_t *thrInstId,* ipIfUniqueId_t *ifId,* ipAddr_t * *pAddr*)

Function used to register border router server address.

Parameters

in	*thrInstId*	Thread instance ID
in	*ifId*	IP Interface identifier
in	*pAddr*	Pointer to the IP address

Returns

uint8_t

6.5.13 void MESHCOP_NwkJoinWithCommissioning (instanceId_t *thrInstId,* thrNwkJoiningEntry_t * *pNwkJoiningList,* uint32_t *nbOfNwkJoiningEntries*)

Run through the pNwkJoiningList list and join using the commissioning procedure. NOTE:

- More callback functions must be registered (using EVM_RegisterStatic() function) with the gThrEvSet_NwkJoin_c and gThrEvSet_NwkCommissioning_c event set to receive the network join events.
- pNwkJoiningList is a allocated buffer and it will be free by the function

Parameters

in	*thrInstID*	Thread instance Id
in	*pNwkJoiningList*	Pointer to network joining list

in	*nbOfNwk↩ JoiningEntries*	Size of network joining list

Returns

thrStatus_t Status

6.5.14 nwkStatus_t MESHCOP_Set (instanceId_t *thrInstId,* uint8_t * *pTlvs,* uint32_t *tlvsLength,* meshcopHandlerCb_t *pfSetCb*)

Function used to do a ManagementSet.

Parameters

in	*thrInstId*	Thread instance ID
in	*pTlvs*	Pointer to the start of the TLVs buffer
in	*tlvsLen*	Length of the TLVs buffer
in	*pfSetCb*	Pointer to the function which will be called when the request is completed

Returns

nwkStatus_t

6.5.15 nwkStatus_t MESHCOP_Get (instanceId_t *thrInstId,* uint8_t * *pTlvs,* uint32_t *tlvsLength,* meshcopHandlerCb_t *pfGetCb*)

Function used to do a management get.

Parameters

in	*thrInstId*	Thread instance ID
in	*pTlvs*	Pointer to the list of TLV IDs
in	*tlvsLen*	Length of the TLV IDs list
in	*pfGetCb*	Pointer to the function which will be called when the request is completed
in	*pNeighborIp↩ Addr*	Pointer to the specific neighbor IP address(optional)

Returns

nwkStatus_t

6.5.16 nwkStatus_t MESHCOP_MgmtSendPanIdQuery (instanceId_t *thrInstId,* uint32_t *channelMask,* uint16_t *panId,* meshcopHandlerCb_t *pfHandlerCb,* ipAddr_t * *plpAddr*)

Request to search for Pan ID conflict.

Parameters

in	*thrInstId*	Thread instance ID
in	*channelMask*	Mask of channels
in	*panId*	Pan ID
in	*pfHandlerCb*	Pointer to the function which will be called when the request is completed
in	*pIpAddr*	Pointer to the IP address of the node which will search for Pan ID conflict

Returns

nwkStatus_t

6.5.17 nwkStatus_t MESHCOP_MgmtSendEdScan (instanceId_t *thrInstId,* uint32_t *channelMask,* uint32_t *count,* uint32_t *period,* uint32_t *scanDuration,* meshcopHandlerCb_t *pfHandlerCb,* ipAddr_t * *pIpAddr*)

Request to do ED scan.

Parameters

in	*thrInstId*	Thread instance ID
in	*channelMask*	Mask of channels
in	*count*	Count
in	*period*	Period
in	*scanDuration*	Scan duration
in	*pfHandlerCb*	Pointer to the function which will be called when the request is completed
in	*pIpAddr*	Pointer to the IP address of the node which will search for Pan ID conflict

Returns

nwkStatus_t

6.5.18 nwkStatus_t MESHCOP_MgmtSendAnnounceBegin (instanceId_t *thrInstId,* uint16_t *commissionerSessionId,* uint32_t *channelMask,* uint32_t *count,* uint16_t *period,* ipAddr_t * *pIpAddr*)

Request to send a MGMT_ANNOUNCE_BEGIN.ntf

Parameters

in	*thrInstId*	Thread instance ID
in	*commissioner↩ SessionId*	Commissioner Session ID
in	*channelMask*	Mask of channels
in	*count*	Count
in	*period*	Period
in	*pIpAddr*	Pointer to the IP address of the node which will begin sending the M↩ GMT_ANNOUNCE.ntf

Returns

nwkStatus_t

6.5.19 nwkStatus_t MESHCOP_MgmtActiveSet (meshcopMgmtParams_t * *pParams*)

Function used to send a MGMT_ACTIVE_SET packet.

Parameters

in	*pParams*	Pointer to the input parameters

Returns

nwkStatus_t

6.5.20 nwkStatus_t MESHCOP_MgmtPendingSet (meshcopMgmtParams_t * *pParams*)

Function used to send a MGMT_PENDING_SET packet.

Parameters

in	*pParams*	Pointer to the input parameters

Returns

nwkStatus_t

6.5.21 nwkStatus_t MESHCOP_MgmtCommGet (meshcopMgmtParams_t * *pParams*)

Function used to send a MGMT_COMMISSIONER_GET packet.

Parameters

in	*pParams*	Pointer to the input parameters

Returns

nwkStatus_t

6.5.22 nwkStatus_t MESHCOP_MgmtActiveGet (meshcopMgmtParams_t * *pParams*)

Function used to send a MGMT_ACTIVE_GET packet.

Parameters

in	*pParams*	Pointer to the input parameters

Returns

nwkStatus_t

6.5.23 nwkStatus_t MESHCOP_MgmtPendingGet (meshcopMgmtParams_t * *pParams*)

Function used to send a MGMT_PENDING_GET packet.

Parameters

in	*pParams*	Pointer to the input parameters

Returns

nwkStatus_t

6.6 Variable Documentation

6.6.1 list_t gThrExpectedJoinerList

List of expected joiners (of type expectedJoinerEntry_t)

6.6.2 thrCommissionerMode_t gMeshcopCommissionerMode

The current commissioner mode.

Chapter 7
Network IP Sockets Interface

7.1 Overview

Files

- file sockets.h

Data Structures

- struct sockaddrIn_t
- struct sockaddrIn6_t
- struct sockaddrStorage_t
- struct timeval
- struct ipMreq_t
- struct socketCallback_t
- struct sock_t

Macros

- #define BSDS_DATAGRAM_SUPPORT
- #define BSDS_STREAM_SUPPORT
- #define BSDS_BLOCKING_SOCKET
- #define BSDS_SELECT_SUPPORT
- #define BSDS_OPTIONS_SUPPORT
- #define BSDS_RECV_EVENT
- #define BSDS_CANCEL_SELECT_EVENT
- #define BSDS_CONN_DONE_EVENT
- #define SOCK_DGRAM
- #define SOCK_STREAM
- #define PF_INET
- #define PF_INET6
- #define MSG_DONTWAIT
- #define MSG_SEND_WITH_MEMBUFF
- #define MSG_GET
- #define IPV6_UNICAST_HOPS
- #define IPV6_MULTICAST_HOPS
- #define IPV6_ADD_MEMBERSHIP
- #define IPV6_DROP_MEMBERSHIP
- #define IPV6_MTU
- #define IPV6_JOIN_ANYCAST
- #define IPV6_TCLASS
- #define IP_TOS
- #define IP_TTL
- #define IP_ADD_MEMBERSHIP
- #define IP_DROP_MEMBERSHIP
- #define IP_MULTICAST_IF

- #define IP_MULTICAST_TTL
- #define IP_MULTICAST_LOOP
- #define IPV6_JOIN_GROUP
- #define IPV6_LEAVE_GROUP
- #define BSDS_DEFAULT_FLOW_FLAGS
- #define **BSDS_SET_TX_SEC_FLAGS**(macKeyIdMode, macSecLevel)
- #define FD_SETSIZE

Typedefs

- typedef uint32_t **socklen_t**

Enumerations

- enum sockFuncErr_t {
  gBsdsSockSuccess_c,
  gBsdsSockUnavailable_c,
  gBsdsSockAdded_c,
  gBsdsSockRemoved_c,
  gBsdsSockListFull_c,
  gBsdsSockError_g,
  gBsdsSockPortInUse_c,
  gBsdsNoMoreFreePorts_c,
  gBsdsSockFound_c,
  gBsdsSockInvalid_c }
- enum sockStateErr_t {
  gBsdsSockUnbound_c,
  gBsdsSockBound_c,
  gBsdsSockListening_c,
  gBsdsSockUnConnected_c,
  gBsdsSockConnected_c }

Functions

- int32_t socket (uint8_t domain, uint8_t type, uint8_t protocol)
- int32_t shutdown (int32_t sockfd, int how)
- int32_t bind (int32_t sockfd, sockaddrStorage_t *pLocalAddr, uint32_t addrlen)
- int32_t send (int32_t sockfd, void *msg, uint32_t msgLen, uint32_t flags)
- int32_t **sendmsg** (int32_t sockfd, ipAddr_t *pSrc, void *msg, uint32_t msgLen, uint32_t flags, sockaddrStorage_t *pTo, socklen_t toLen)
- int32_t sendto (int32_t sockfd, void *msg, uint32_t msgLen, uint32_t flags, sockaddrStorage_t *p↩ To, uint32_t toLen)
- int32_t recv (int32_t sockfd, void *msg, uint32_t msgLen, uint32_t flags)
- int32_t recvfrom (int32_t sockfd, void *msg, uint32_t msgLen, uint32_t flags, sockaddrStorage_t *from, socklen_t *fromLen)
- int32_t connect (int32_t sockfd, sockaddrStorage_t *serv_addr, uint32_t addrLen)
- int32_t getsockopt (int32_t sockfd, int32_t level, int32_t optName, void *optVal, int32_t *optLen)
- int32_t setsockopt (int32_t sockfd, int32_t level, int32_t optName, void *optVal, int32_t optLen)
- int32_t getsockname (int32_t sockfd, sockaddrStorage_t *pAddr, socklen_t *addrlen)
- sock_t * getsocket (int32_t sockFd)

Variables

- const socketCallback_t **sockDgramCallback**
- const socketCallback_t **sockStreamCallback**

7.2 Data Structure Documentation

7.2.1 struct sockaddrIn_t

Data Fields

ipAddr_t	sin_addr	Internet address.
uint16_t	sin_family	Address family.
uint16_t	sin_port	Port number.

7.2.2 struct sockaddrIn6_t

Data Fields

ipAddr_t	sin6_addr	IPV6 address.
uint16_t	sin6_family	The address family we used when we set up the socket (AF_INET6)
uint16_t	sin6_port	The port number (the transport address)
uint32_t	sin6_flowinfo	IPV6 flow information (LSB= (MAC key id mode) | (MAC security level))
uint32_t	sin6_scope_id	set of interfaces for a scope (RFC2553) or media interface handle

7.2.3 struct sockaddrStorage_t

Data Fields

ipAddr_t	ss_addr	Internet address.
uint16_t	ss_family	Address family.
uint8_t	__data[sizeof(uint16_t)+sizeof(uint32_t)+sizeof(uint32_t)]	Storage large enough and aligned for storing the socket address data structure of any family.

7.2.4 struct timeval

Structure containing time information.

Data Fields

uint32_t	tv_sec	seconds
uint32_t	tv_usec	microseconds

7.2.5 struct ipMreq_t

Data Fields

ipAddr_t	imrMultiaddr	IP multicast group address.
ipAddr_t	imrInterface	IP address of local interface.

7.2.6 struct socketCallback_t

Data Fields

- int32_t(∗ SocketBind)(int32_t sockfd, sockaddrStorage_t ∗pLocalAddr, uint32_t addrlen)
- int32_t(∗ SocketConnect)(int32_t sockfd, sockaddrStorage_t ∗serv_addr, int addrLen)
- int32_t(∗ SocketListen)(int32_t sockfd, uint32_t backlog)
- int32_t(∗ SocketAccept)(int32_t sockfd, sockaddrStorage_t ∗addr, int addrLen)
- int32_t(∗ SocketRecv)(int32_t sockfd, void ∗msg, uint32_t msgLen, uint32_t flags)
- int32_t(∗ SocketRecvFrom)(int32_t sockfd, void ∗msg, uint32_t msgLen, uint32_t flags, sockaddrStorage_t ∗from, socklen_t fromLen)
- int32_t(∗ SocketSend)(int32_t sockfd, void ∗msg, uint32_t msgLen, uint32_t flags)
- int32_t(∗ SocketSendto)(int32_t sockfd, ipAddr_t ∗pAddr, void ∗msg, uint32_t msgLen, uint32_t flags, sockaddrStorage_t ∗to, socklen_t toLen)
- int32_t(∗ **SocketClose**)(int32_t sockfd, int how)

7.2.6.1 Field Documentation

7.2.6.1.1 int32_t(∗ socketCallback_t::SocketBind) (int32_t sockfd, sockaddrStorage_t ∗pLocalAddr, uint32_t addrlen)

< Socket bind callback

Socket connect (TCP) callback

7.2.6.1.2 int32_t(∗ socketCallback_t::SocketConnect) (int32_t sockfd, sockaddrStorage_t ∗serv_addr, int addrLen)

Socket listen (TCP) callback.

7.2.6.1.3 int32_t(∗ socketCallback_t::SocketListen) (int32_t sockfd, uint32_t backlog)

Socket accept (TCP) callback.

7.2.6.1.4 int32_t(∗ socketCallback_t::SocketAccept) (int32_t sockfd, sockaddrStorage_t ∗addr, int addrLen)

Socket recv (TCP) callback.

7.2.6.1.5 int32_t(∗ socketCallback_t::SocketRecv) (int32_t sockfd, void ∗msg, uint32_t msgLen, uint32_t flags)

Socket recvfrom (UDP) callback.

7.2.6.1.6 int32_t(∗ socketCallback_t::SocketRecvFrom) (int32_t sockfd, void ∗msg, uint32_t msgLen, uint32_t flags, sockaddrStorage_t ∗from, socklen_t fromLen)

Socket send (TCP) callback.

7.2.6.1.7 int32_t(∗ socketCallback_t::SocketSend) (int32_t sockfd, void ∗msg, uint32_t msgLen, uint32_t flags)

Socket sendto (UDP) callback.

7.2.6.1.8 int32_t(∗ socketCallback_t::SocketSendto) (int32_t sockfd, ipAddr_t ∗pAddr, void ∗msg, uint32_t msgLen, uint32_t flags, sockaddrStorage_t ∗to, socklen_t toLen)

Socket close (UDP/TCP) callback.

7.2.7 struct sock_t

Data Fields

uint8_t	addrFam	Address family.
uint8_t	prot	Protocol.
uint8_t	type	Socket type(datagram or stream)
uint8_t	state	Socket status of the connection.

uint8_t	flags	Socket flags.
uint8_t	tspConnIndex	Transport connection index.
uint8_t	sockPad	padding
socketCallback_t *	pCallback	Pointer to socket callbacks.

7.3 Macro Definition Documentation

7.3.1 #define BSDS_DATAGRAM_SUPPORT

Support datagram sockets(using UDP)

7.3.2 #define BSDS_STREAM_SUPPORT

Support stream sockets(using TCP)

7.3.3 #define BSDS_BLOCKING_SOCKET

Support blocking sockets.

7.3.4 #define BSDS_SELECT_SUPPORT

Sockets module support select functionality.

7.3.5 #define BSDS_OPTIONS_SUPPORT

Support socket options.

7.3.6 #define BSDS_RECV_EVENT

Event to be used for Socket receive blocking.

7.3.7 #define BSDS_CANCEL_SELECT_EVENT

Event to be used for Socket select blocking.

7.3.8 #define BSDS_CONN_DONE_EVENT

Event to be used for receiving a connection.

7.3.9 #define SOCK_DGRAM

Datagram socket type.

7.3.10 #define SOCK_STREAM

Stream socket type.

7.3.11 #define PF_INET

IPv4 family.

7.3.12 #define PF_INET6

IPv6 family.

7.3.13 #define MSG_DONTWAIT

Nonblocking IO.

7.3.14 #define MSG_SEND_WITH_MEMBUFF

socket send app data is located in a memory buffer allocated from memory pools and will be freed by the stack

7.3.15 #define MSG_GET

Message get.

7.3.16 #define IPV6_UNICAST_HOPS

Set the unicast hop limit for the socket.

7.3.17 #define IPV6_MULTICAST_HOPS

Set the multicast hop limit for the socket.

7.3.18 #define IPV6_ADD_MEMBERSHIP

Joins the IPv6 multicast group specified.

7.3.19 #define IPV6_DROP_MEMBERSHIP

Leaves the IPv6 multicast group specified.

7.3.20 #define IPV6_MTU

Set/Get MTU only for connected socket.

7.3.21 #define IPV6_JOIN_ANYCAST

Joins the anycast group specified.

7.3.22 #define IPV6_TCLASS

Set the traffic class field of IPv6 header.

7.3.23 #define IP_TOS

Sets the TOS field from IPv4 header.

7.3.24 #define IP_TTL

Sets the Time To Live (TTL) in the IP header for unicast packets.

7.3.25 #define IP_ADD_MEMBERSHIP

Joins the multicast group specified.

7.3.26 #define IP_DROP_MEMBERSHIP

Leaves the multicast group specified.

7.3.27 #define IP_MULTICAST_IF

Sets the interface over which outgoing multicast datagrams are sent.

7.3.28 #define IP_MULTICAST_TTL

Sets the Time To Live (TTL) in the IP header for outgoing multicast datagrams.

7.3.29 #define IP_MULTICAST_LOOP

Specifies whether or not a copy of an outgoing multicast datagram is delivered to the sending host as long as it is a member of the multicast group.

7.3.30 #define IPV6_JOIN_GROUP

Joins the IPv6 multicast group specified.

7.3.31 #define IPV6_LEAVE_GROUP

Leaves the IPv6 multicast group specified.

Get default socket flow flag

7.3.32 #define BSDS_DEFAULT_FLOW_FLAGS

Set MAC security flags.

7.3.33 #define FD_SETSIZE

File descriptor list size to select/poll on.

7.4 Enumeration Type Documentation

7.4.1 enum sockFuncErr_t

Enumerator

gBsdsSockSuccess_c Success.
gBsdsSockUnavailable_c Socket to be removed is not available.
gBsdsSockAdded_c Socket was successfully added to list.
gBsdsSockRemoved_c Socket was successfully removed from list.
gBsdsSockListFull_c The sockets list is full.
gBsdsSockError_g Error.
gBsdsSockPortInUse_c Port was already used.
gBsdsNoMoreFreePorts_c There are no more free ports.
gBsdsSockFound_c Socket was found in list.
gBsdsSockInvalid_c Invalid.

7.4.2 enum sockStateErr_t

Enumerator

gBsdsSockUnbound_c Socket is not in use.
gBsdsSockBound_c Socket is bound to an address/port combination.
gBsdsSockListening_c Socket is in listening state.
gBsdsSockUnConnected_c Socket is not connected.
gBsdsSockConnected_c Socket is connected.

7.5 Function Documentation

7.5.1 int32_t socket (uint8_t *domain,* uint8_t *type,* uint8_t *protocol*)

This function creates a socket structure(and initialize its values with default) using a specific domain, type and protocol.

Parameters

in	*domain*	Domain which can be PF_INET or PF_INET6
in	*type*	Type of socket(SOCK_DGRAM or SOCK_STREAM)
in	*protocol*	Transport protocol to be used(IPPROTO_UDP or IPPROTO_TCP)

Returns

int32_t Socket file descriptor or -1 in case of error

7.5.2 int32_t shutdown (int32_t *sockfd,* int *how*)

This function close a socket connection.

Parameters

in	*sockfd*	Socket descriptor
in	*how*	(UNUSED)parameter which specifies how the socket will be closed

Returns

0 On success
-1 On failure

7.5.3 int32_t bind (int32_t *sockfd,* sockaddrStorage_t * *pAddr,* uint32_t *addrLen*)

Public interface function for Sockets module. This function is used to bind a local IP address and a local port to an existing socket.

Parameters

in	*sockfd*	Socket descriptor
in	*pAddr*	Pointer to the socket address structure
in	*addrLen*	Size of the pAddr structure

Returns

0 On success
-1 On failure

7.5.4 int32_t send (int32_t *sockfd,* void * *msg,* uint32_t *msgLen,* uint32_t *flags*)

This function is used to send data to a connected socket.

Parameters

in	*sockfd*	Socket descriptor
in	*msg*	Pointer to the data which needs to be send
in	*msgLen*	Length of the data which needs to be send
in	*flags*	Flags used for sending

Returns

int32_t Length of the data sent, -1 on failure

7.5.5 int32_t sendto (int32_t *sockfd,* void * *msg,* uint32_t *msgLen,* uint32_t *flags,* sockaddrStorage_t * *pTo,* uint32_t *toLen*)

This function is used to send data to a specific socket.

Parameters

in	*sockfd*	Socket descriptor
in	*msg*	Pointer to the data which needs to be send
in	*msgLen*	Length of the data which needs to be send
in	*flags*	Flags used for sending
in	*pTo*	Pointer to the remote socket address structure
in	*toLen*	Size of the remote address structure

Returns

int32_t Length of the data sent, -1 on failure

7.5.6 int32_t recv (int32_t *sockfd,* void * *msg,* uint32_t *msgLen,* uint32_t *flags*)

This function is used to get data from a socket RX queue.

Parameters

in	*sockfd*	Socket descriptor
out	*msg*	Pointer to the buffer responsible for holding received data
in	*msgLen*	Length of the buffer allocated for receiving data
in	*flags*	Flags used for receiving

Returns

int32_t Length of the data received, -1 on failure

7.5.7 int32_t recvfrom (int32_t *sockfd,* void * *msg,* uint32_t *msgLen,* uint32_t *flags,* sockaddrStorage_t * *from,* socklen_t * *fromLen*)

This function is used to get data from a specific socket from the RX queue. The remote information will be placed in the from structure.

Parameters

in	*sockfd*	Socket descriptor
out	*msg*	Pointer to the buffer responsible for holding received data
in	*msgLen*	Length of the buffer allocated for receiving data

in	*flags*	Flags used for receiving
out	*from*	Pointer to the remote socket address structure
in	*fromLen*	Pointer to the size of the remote address structure

Returns

int32_t Length of the data received, -1 on failure

7.5.8 int32_t connect (int32_t *sockfd,* sockaddrStorage_t * *serv_addr,* uint32_t *addrLen*)

This function is used to connect to a remote server.

Parameters

in	*sockfd*	Socket descriptor
in	*serv_addr*	Address structure for the server to connect to
in	*addrLen*	Address structure length

Returns

0 On success
-1 On error

7.5.9 int32_t getsockopt (int32_t *sockfd,* int32_t *level,* int32_t *optName,* void * *optVal,* int32_t * *optLen*)

This function retrieves information about a specified socket.

Parameters

in	*sockfd*	Socket file descriptor
in	*level*	Layer for operation
in	*optName*	Option
out	*optVal*	Pointer to the value for the option
out	*optLen*	Pointer to the length of the option

Return values

gBsdsSockSuccess_c	If the option was set
gBsdsSockError_g	If the option cannot be set

7.5.10 int32_t setsockopt (int32_t *sockfd,* int32_t *level,* int32_t *optName,* void * *optVal,* int32_t *optLen*)

This function sets information for a specified socket.

Parameters

in	*sockfd*	Socket file descriptor
in	*level*	Layer for operation
in	*optName*	Option
in	*optVal*	Pointer to the value for the option
in	*optLen*	The length of the option

Return values

gBsdsSockSuccess_c	If the option was set
gBsdsSockError_g	If the option cannot be set

7.5.11 int32_t getsockname (int32_t *sockfd,* sockaddrStorage_t * *pAddr,* socklen_t * *addrlen*)

This function retrieves information about the local address and port of a socket.

Parameters

in	*sockfd*	Socket file descriptor
out	*pAddr*	A pointer to a structure containing the local information
out	*addrlen*	Pointer to the size of the pAddr structure

Return values

gBsdsSockSuccess_c	If the name can be retrieved
gBsdsSockError_g	If the name cannot be retrieved

7.5.12 sock_t * getsocket (int32_t *sockFd*)

This function gets a pointer to the socket structure.

Parameters

in	*sockFd*	Socket descriptor

Returns

sock_t* Pointer to the socket structure or NULL if it doesn't exist

Chapter 8
CoAP Interface

8.1 Overview

Files

- file coap.h
- file coap_cfg.h

Data Structures

- struct coapUriPath_t
- struct coapInstance_t
- struct coapCallbackStruct_t
- struct coapTokenCbStruct_t
- struct coapOptionDetails_t
- struct coapSession_t
- struct coapStartSecParams_t
- struct coapStartUnsecParams_t
- struct coapRegCbParams_t

Macros

- #define COAP_ENABLED
- #define COAP_MAX_URI_PATH_OPT_SIZE
- #define COAP_MAX_OPTION_VALUE_SIZE
- #define COAP_MAX_TOKEN_LEN
- #define COAP_OBSERVE_OPTION
- #define **COAP_URI_PATH_OPTION**
- #define **COAP_CONTENT_FORMAT_OPTION**
- #define **COAP_URI_QUERY_OPTION**
- #define **COAP_ACCEPT_OPTION**
- #define **COAP_TIMER_INTERVAL**
- #define COAP_DEFAULT_PORT
- #define **COAP_DEFAULT_SECURED_PORT**
- #define COAP_INSTANCES_URI_PATH
- #define COAP_SetMaxRetransmitCount(coapInstanceId, maxRetransmitCount)
- #define COAP_MAX_CALLBACKS
- #define COAP_MAX_NON_PIGGYBACKED_RSP
- #define **COAP_MAX_INSTANCES**
- #define COAP_MAX_MSG_IDs
- #define COAP_MAX_OPTIONS
- #define COAP_TOKEN_LENGTH

Typedefs

- typedef void(∗ coapCallback_t) (coapSessionStatus_t sessionStatus, void ∗pData, struct coap↩ Session_tag ∗pSession, uint32_t dataLen)

- typedef coapOptionDetails_t **coapOption_t**

Enumerations

- enum coapSessionStatus_t {
  gCoapSuccess_c,
  gCoapFailure_c,
  gCoapDuplicate_c }
- enum coapMacSecFlags_t {
  gCoapMacSecMode0Level5_c,
  gCoapMacSecMode1Level5_c,
  gCoapMacSecUnsecured_c }
- enum coapMsgTypesAndCodes_t {
  gCoapMsgTypeConPost_c,
  gCoapMsgTypeNonPost_c,
  gCoapMsgTypeAckSuccessChanged_c,
  gCoapMsgTypeAckSuccessContent_c,
  gCoapMsgTypeConGet_c,
  gCoapMsgTypeNonGet_c,
  gCoapMsgTypeEmptyAck_c,
  gCoapMsgTypeUseSessionValues_c }
- enum coapMessageTypes_t {
  gCoapConfirmable_c,
  gCoapNonConfirmable_c,
  gCoapAcknowledgement_c,
  gCoapReset_c }
- enum coapReqRespCodes_t {

gCoapGET_c,
gCoapPOST_c,
gCoapPUT_c,
gCoapDELETE_c,
gEmpty_c,
gCreated_c,
gDeleted_c,
gValid_c,
gChanged_c,
gContent_c,
gBadRequest_c,
gUnauthorized_c,
gBadOption_c,
gForbidden_c,
gNotFound_c,
gMethodNotAllowed_c,
gNotAcceptable_c,
gPreconditionFailed_c,
gRequestEntityTooLarge_c,
gUnsupportedContentFormat_c,
gInternalServerError_c,
gNotImplemented_c,
gBadGateway_c,
gServiceUnavailable_c,
gGatewayTimeout_c,
gProxyingNotSupported_c }

Functions

- void COAP_Init (taskMsgQueue_t *pTaskMsgQueue)
- uint8_t COAP_CreateInstance (coapStartSecParams_t *pCoapStartSecParams, coapStartUnsec↩ Params_t *pCoapStartUnsecParams, ipIfUniqueId_t ipIfId, coapRegCbParams_t *pCallbacks↩ Struct, uint32_t nbOfCallbacks)
- bool_t COAP_CloseInstance (uint8_t coapInstanceId)
- coapSession_t * COAP_OpenSession (uint8_t coapInstanceId)
- void COAP_CloseSession (coapSession_t *pSession)
- void COAP_AddOptionToList (coapSession_t *pSession, uint8_t optName, uint8_t *optValue, uint8_t optValueLen)
- void COAP_SetUriPath (coapSession_t *pSess, coapUriPath_t *pUriPath)
- nwkStatus_t COAP_Send (coapSession_t *pSession, coapMsgTypesAndCodes_t coapMsgType, void *pData, uint32_t payloadLen)
- nwkStatus_t COAP_SendMsg (coapSession_t *pSession, void *pData, uint32_t payloadLen)
- nwkStatus_t COAP_RegisterResourceCallback (uint8_t coapInstanceId, coapRegCbParams_t *p↩ CallbacksStruct, uint32_t nbOfCallbacks)
- void COAP_RegisterTokenCallback (coapSession_t *pSession, coapCallback_t pCallback)
- bool_t COAP_UnregisterTokenCallback (coapSession_t *pSession, coapCallback_t pCallback)
- bool_t COAP_UnregisterResourceCallback (uint8_t coapInstanceId, coapRegCbParams_t *p↩ CallbacksStruct, uint32_t nbOfCallbacks)

- void COAP_CloseAnySession (void)

8.2 Data Structure Documentation

8.2.1 struct coapUriPath_t

URI-path structure of a CoAP message.

Data Fields

uint8_t	length	sizeof URI-path. e.g. for "/thread/client", uriPathLen = 14
uint8_t *	pUriPath	pointer to URI-path. URI-path example: "/thread/client"

8.2.2 struct coapInstance_t

This structure keeps the parameters of a CoAP instance.

Data Fields

void *	pTransport	sockFd or DTLS peer/context(for servers)
list_t	sessionList	list of ongoing CoAP sessions
uint16_t	port	UDP port number.
uint16_t	coapAck↩ TimeoutMs	the minimum spacing before another retransmission in milliseconds
ipIfUniqueId↩ _t	ipIfId	IP interface unique ID.
uint8_t	coapMax↩ Retransmit	number of retransmissions
bool_t	usedEntry	TRUE - if entry is populated, FALSE - entry is free.

8.2.3 struct coapCallbackStruct_t

The URI-path, callback and instance id tuple for associating an incoming message with its callback.

Data Fields

const coapUriPath_t *	pUriPathStruct	pointer to URI-path and its length

coapCallback_t	pCallback	pointer to callback function
uint8_t	coapInstanceId[COAP_INSTANCES_URI_PATH]	CoAP instance ID array.

8.2.4 struct coapTokenCbStruct_t

The token, callback, instance id tuple needed for associating a non-piggybacked message with its callback.

Data Fields

uint8_t	aToken[COAP_MAX_TOKEN_LEN]	Token of variable length.
coapCallback_t	pCallback	pointer to callback function
uint8_t	coapInstanceId	CoAP instance ID.
uint8_t	tokenLen	The length of the token.

8.2.5 struct coapOptionDetails_t

CoAP option structure.

Data Fields

uint8_t	optName	The ID of the CoAP option.
uint8_t	optValueLen	Length in bytes of the option value.
uint8_t	optValue[COAP_MAX_OPTION_VALUE_SIZE]	Option value.

8.2.6 struct coapSession_t

A CoAP session keeps all the information necessary for a CoAP message exchange.

Data Fields

ipAddr_t	remoteAddr	The destination IP address.
ipAddr_t	sourceAddr	The source IP address.
coapOption_t *	pTxOptionList	Options to be included in an outgoing CoAP msg.
coapOption_t *	pRxOptionList	Options received in the incoming CoAP msg.
coapCallback_t	pCallback	Pointer to callback function.
uint8_t	aToken[COAP_MAX_TOKEN_LEN]	Token of variable length.
coapUriPath_t *	pUriPath	Pointer to URI-Path structure.
uint32_t	macTxFlags	Sets the security level and the key ID mode for MAC layer encryption. This value is by default set to key ID mode 1 and encryption level 5 and should not be modified
uint16_t	messageID	Message ID is a random number generated by CoAP module.
uint16_t	remotePort	Source port.
ipIfUniqueId_t	ipIfId	IP interface id.
uint8_t	coapInstId	CoAP instance ID.
uint8_t	tokenLen	The length of the token.
bool_t	isDtlsSecured	A flag indicating if the message uses a secure connection (DTLS) or not.
bool_t	autoClose	Set this flag to FALSE if the coap session should not be automatically closed when receiving the ACK (on the requester) or sending the ACK (on the server)
bool_t	bIsSubscribed	On client side keeps record if the server successfully registered the client as observer and pass this parameter to application. On server side, application checks this parameter to see if the client asked for subscription and depending on the server availability responds positive or not.
uint8_t	observeOption	Value of the observe option. LSB 24 bits keep the sequence id
coapMessageTypes_t	msgType	CoAP message types are: CON, NON, ACK, RESET.
coapReqRespCodes_t	code	Depending on the message type (Request or Response)
uint8_t	lastHopLQI	LQI of last hop, if the message is multi hop.
uint8_t	hopLimit	Hop limt to use when sending a COAP packet.
uint8_t	ipQos	Ip packet Quality of service -> DSCP field.

8.2.7 struct coapStartSecParams_t

Parameters needed for creating a secured CoAP instance over DTLS.

Data Fields

sockaddr↩ Storage_t *	pServerAddr	DTLS Server's IP address.
sockaddr↩ Storage_t *	pLocalAddr	My local IP address.
uint32_t	retransmit↩ TimeUnits	Number of time units to retransmit a packet.
uint8_t	max↩ RetransmitCnt	Number of message retransmissions.

8.2.8 struct coapStartUnsecParams_t

Parameters needed for creating an unsecured CoAP instance over UDP.

Data Fields

uint16_t	port	UDP port.
uint8_t	addrFamily	address family (AF_INET and AF_INET6 supported)

8.2.9 struct coapRegCbParams_t

The callback, URI-path tuple for associating an incoming message with its callback function.

Data Fields

coapCallback↩ _t	pCallback	pointer to the callback function
coapUriPath_t *	pUriPathStruct	pointer to URI-path and its length

8.3 Macro Definition Documentation

8.3.1 #define COAP_ENABLED

Macro used to disable or enable Coap when compiling the Thread library.

8.3.2 #define COAP_MAX_URI_PATH_OPT_SIZE

The maximum length of URI-path options.

8.3.3 #define COAP_MAX_OPTION_VALUE_SIZE

Maximum length of one option.

URI-path options are not limited by this value if they are added with COAP_SetUriPath() function

8.3.4 #define COAP_MAX_TOKEN_LEN

Maximum length of token as defined in RFC 7252.

8.3.5 #define COAP_OBSERVE_OPTION

CoAP Option Names.

8.3.6 #define COAP_DEFAULT_PORT

CoAP Ports.

8.3.7 #define COAP_INSTANCES_URI_PATH

Number of CoAP instances allowed for one URI-path.

8.3.8 #define COAP_SetMaxRetransmitCount(*coapInstanceId, maxRetransmitCount*)

Set number of retransmissions for a CON message.

8.3.9 #define COAP_MAX_CALLBACKS

< Maximum number of callbacks

Maximum number of callbacks registered for non-piggybacked responses

8.3.10 #define COAP_MAX_NON_PIGGYBACKED_RSP

Maximum number of active CoAP sessions at a give moment, per one CoAP instance.

Maximum number of CoAP instances

8.3.11 #define COAP_MAX_MSG_IDs

Used for keeping track of duplicate CoAP messages.

8.3.12 #define COAP_MAX_OPTIONS

Maximum number of options included in one CoAP message.

Here are not included URI-path options which MAY be added with COAP_SetUriPath() function

8.3.13 #define COAP_TOKEN_LENGTH

Token length used by default in CoAP messages.

8.4 Typedef Documentation

8.4.1 typedef void(∗ coapCallback_t) (coapSessionStatus_t sessionStatus, void ∗pData, struct coapSession_tag ∗pSession, uint32_t dataLen)

CoAP callback function prototype for receiving a CoAP message.

8.5 Enumeration Type Documentation

8.5.1 enum coapSessionStatus_t

Return status of a CoAP session.

Enumerator

gCoapSuccess_c CoAP transaction succeeded.
gCoapFailure_c Retransmission timer expired and no reply was received.
gCoapDuplicate_c A message with same message ID was received in the latest gCoapMaxMsgIds messages.

8.5.2 enum coapMacSecFlags_t

Security at MAC layer for CoAP messages.

By default, all messages use gCoapMacSecMode1Level5_c

8.5.3 enum coapMsgTypesAndCodes_t

This enum is meant to compress the most used message type and code combinations in one constant.

Enumerator

gCoapMsgTypeConPost_c CON POST message.
gCoapMsgTypeNonPost_c NON POST message.
gCoapMsgTypeAckSuccessChanged_c ACK Success Changed message.
gCoapMsgTypeAckSuccessContent_c ACK Success Content message.
gCoapMsgTypeConGet_c CON GET message.
gCoapMsgTypeNonGet_c NON GET message.
gCoapMsgTypeEmptyAck_c ACK Empty message.
gCoapMsgTypeUseSessionValues_c Use the (msgType, code) values set in the session.

8.5.4 enum coapMessageTypes_t

CoAP message types.

8.5.5 enum coapReqRespCodes_t

CoAP Method and Response Codes.

8.6 Function Documentation

8.6.1 void COAP_Init (taskMsgQueue_t * *pTaskMsgQueue*)

This function initializes the CoAP module.

Parameters

in	*pTaskMsg↩ Queue*	Pointer to message task queue.

8.6.2 uint8_t COAP_CreateInstance (coapStartSecParams_t * *pCoapStart↩ SecParams,* coapStartUnsecParams_t * *pCoapStartUnsecParams,* ipIfUniqueId_t *ipIfId,* coapRegCbParams_t * *pCallbacksStruct,* uint32_t *nbOfCallbacks*)

This function opens a secure or unsecured Coap instance.

Parameters

in	*pCoapStart↩ SecParams*	Pointer to initialization structure for a secure transmission over DTLS.
in	*pCoapStart↩ UnsecParams*	Pointer to initialization structure for an unsecured transmission over sockets.
in	*ipIfUniqueId*	ip interface ID.
in	*pCallbacks↩ Struct*	Pointer to callbacks registered for that instance.
in	*nbOfCallbacks*	Number of registered callbacks.

Returns

uint8_t CoAP instance id.

8.6.3 bool_t COAP_CloseInstance (uint8_t *coapInstanceId*)

This function closes a CoAP instance. Make sure that no other module uses the same instance.

Parameters

in	*coapInstanceId*	CoAP instance

Returns

bool_t TRUE - if the closing succeeded FALSE - otherwise

8.6.4 coapSession_t ∗ COAP_OpenSession (uint8_t *coapInstanceId*)

This function opens a CoAP session for a specific instance. A session is identified by the message ID of the message.

Parameters

in	*coapInstanceId*	CoAP instance Id.

Returns

coapSession_t∗ Pointer to CoAP session.

8.6.5 void COAP_CloseSession (coapSession_t ∗ *pSession*)

This function deletes a CoAP session when completed. This function must be called when a response message was received (in the case of the client/initiator), or when a response message is sent (in the case of the server)

Parameters

in	*pSession*	Pointer to CoAP session to be deleted.

8.6.6 void COAP_AddOptionToList (coapSession_t * *pSession*, uint8_t *optName*, uint8_t * *optValue*, uint8_t *optValueLen*)

This function adds the options named by application to a list.

Parameters

in	*pSession*	Pointer to CoAP session.
in	*optName*	The name of the uri-path.
in	*optValue*	The value of the option.
in	*optValueLen*	The length of the option value.

8.6.7 void COAP_SetUriPath (coapSession_t * *pSess*, coapUriPath_t * *pUriPath*)

This function adds the URI-paths to the option list.

Parameters

in	*pSession*	Pointer to CoAP session.
in	*pUriPath*	Pointer to URI-path.

8.6.8 nwkStatus_t COAP_Send (coapSession_t * *pSession*, coapMsg↩ TypesAndCodes_t *coapMsgType*, void * *pData*, uint32_t *payloadLen*)

This function builds and transmits a CoAP message. This function shall be used when sending one of the predefined messages. For a custom build message please use COAP_SendMsg() function.

Parameters

in	*pSession*	Pointer to CoAP session.
in	*coapMsgType*	CoAP message type. Is one from the list coapMsgTypesAndCodes_t.
in	*pData*	Pointer to data payload.
in	*payloadLen*	Payload length.

Returns

nwkStatus_t Status of the operation.

8.6.9 nwkStatus_t COAP_SendMsg (coapSession_t * *pSession*, void * *pData*, uint32_t *payloadLen*)

This function builds and transmits a CoAP message.

Parameters

in	*pSession*	Pointer to CoAP session.
in	*pData*	Pointer to data payload.
in	*payloadLen*	Payload length.

Returns

nwkStatus_t Status of the operation.

8.6.10 nwkStatus_t COAP_RegisterResourceCallback (uint8_t *coapInstanceId*, coapRegCbParams_t * *pCallbacksStruct*, uint32_t *nbOfCallbacks*)

This function registers a callback for a given uri-path name.

Parameters

in	*coapInstanceId*	CoAP instance.
in	*pCallbacks↩ Struct*	Pointer to callback functions array.
in	*nbOfCallbacks*	Number of callbacks.

Returns

nwkStatus_t - Success if registering succeeded
- Fail if table is full

8.6.11 void COAP_RegisterTokenCallback (coapSession_t * *pSession*, coapCallback_t *pCallback*)

This function registers a callback for a given token, for non-piggybacked responses. The client calls this function when it expects another message with the same token.

Parameters

in	*pSession*	Pointer to CoAP session.
in	*pCallback*	Pointer to callback function.

8.6.12 bool_t COAP_UnregisterTokenCallback (coapSession_t * *pSession*, coapCallback_t *pCallback*)

This function unregisters a callback for a given token.

Parameters

in	*pSession*	Pointer to CoAP session.
in	*pCallback*	Pointer to callback function.

Returns

bool_t TRUE - if the unregister succeeded FALSE - otherwise

8.6.13 bool_t COAP_UnregisterResourceCallback (uint8_t *coapInstanceId,* coapRegCbParams_t * *pCallbacksStruct,* uint32_t *nbOfCallbacks*)

This function unregisters a callback for a given uri-path name.

Parameters

in	*coapInstanceId*	CoAP instance.
in	*pCallback*	Pointer to callback functions array.
in	*nbOfCallbacks*	Number of callbacks.

Returns

bool_t TRUE - if the unregister succeeded FALSE - otherwise

8.6.14 void COAP_CloseAnySession (void)

This function close any sessions for CoAP module.

Chapter 9
Network IP Interface

9.1 Overview

Files

- file ip_if_management.h

Data Structures

- struct ip4IfStruct_t
- struct ip6IfStruct_t
- struct mediaIfStruct_t
- struct ipIfStruct_t
- struct ip4IfAddrData_t
- struct ip6IfAddrData_t

Macros

- #define **IP_IF_MAC_ADDR_NB**

Typedefs

- typedef ipIfStruct_t * ifHandle_t
- typedef void(* ip6IfSelThreadMLSrcAddr6_t) (ipAddr_t *pDestAddr, ipAddr_t **pBestSourceAddr)

Enumerations

- enum ip6AddrType_t {
  ip6AddrTypeManual_c,
  ip6AddrTypeAutoconfigurable_c,
  ip6AddrTypeAutoconfigurableMac2_c }

Functions

- void IP_IF_Init (void)
- uint32_t IP_IF_Add (ipIfUniqueId_t ifId, void *driverHandle, mediaIfStruct_t *pIfStruct, uint16_t ipVersEnabled)
- ifHandle_t IP_IF_GetIfHandle (ipIfUniqueId_t ifId)
- int32_t IP_IF_GetIfIndex (ipIfUniqueId_t ipIfId)
- bool_t IP_IF_IsMyAddr (ipIfUniqueId_t ipIfId, ipAddr_t *pIpAddr)
- void IP_IF_Join (ipIfUniqueId_t ipIfId, ipAddr_t *groupIp)
- void IP_IF_Leave (ipIfUniqueId_t ipIfId, ipAddr_t *groupIp)

- ipIfUniqueId_t IP_IF_GetIfIdByIndex (uint32_t ifIndex)
- ifHandle_t IP_IF_GetIfByIndex (uint32_t ifIndex)
- ifHandle_t IP_IF_GetIfByAddr (ipAddr_t *pIpAddr)

Variables

- void(* **ip4IfStruct_t::ip4Forward**)(ipPktInfo_t *)
- uint32_t ip6IfStruct_t::scope_id
- void ** ip6IfStruct_t::ppNdCfg
- bool_t(* ip6IfStruct_t::ip6IsAddrOnLink)(ipAddr_t *pIpDestAddr, struct ipIfStruct_tag *instanceId)
- bool_t(* ip6IfStruct_t::ip6ResolveUnicastAddr)(ipPktInfo_t *pIpDestAddr)
- void(* ip6IfStruct_t::ip6UpperMgtLayerCb)(ipPktInfo_t *pIpDestAddr)
- uint32_t(* ip6IfStruct_t::ip6McastForward)(ipPktInfo_t *, uint8_t, ipAddr_t *)
- ipAddr_t *(* ip6IfStruct_t::ip6UnicastForward)(ipPktInfo_t *, uint8_t)
- uint32_t(* mediaIfStruct_t::ifOpen)(struct ipIfStruct_tag *)
- uint32_t(* mediaIfStruct_t::ifClose)(struct ipIfStruct_tag *)
- uint32_t(* mediaIfStruct_t::ifSend4)(ipPktInfo_t *)
- uint32_t(* mediaIfStruct_t::ifSendArp)(ipPktInfo_t *, llAddr_t *)
- uint32_t(* mediaIfStruct_t::ifSend6)(ipPktInfo_t *)
- uint32_t(* mediaIfStruct_t::ifGetIID)(struct ipIfStruct_tag *, llAddr_t *, ipAddr_t *)
- uint32_t(* mediaIfStruct_t::ifJoin)(struct ipIfStruct_tag *, ipAddr_t *, uint16_t)
- uint32_t(* mediaIfStruct_t::ifLeave)(struct ipIfStruct_tag *, ipAddr_t *, uint16_t)
- void * ipIfStruct_t::ifDriverHandle
- mediaIfStruct_t * ipIfStruct_t::ifFunctions
- uint16_t ipIfStruct_t::ifMtu
- uint8_t ipIfStruct_t::ipVersion4
- uint8_t ipIfStruct_t::ipVersion6
- llAddr_t ipIfStruct_t::ifDevAddrTbl [IP_IF_MAC_ADDR_NB]
- ipIfUniqueId_t ipIfStruct_t::ifUniqueId
- uint8_t ipIfStruct_t::ifMetric
- ipIfUniqueId_t ip4IfAddrData_t::ipIfId
- uint32_t ip4IfAddrData_t::ip4Addr
- uint32_t ip4IfAddrData_t::ip4SubnetMask
- uint32_t ip4IfAddrData_t::ip4DefaultGw
- ipAddr_t ip6IfAddrData_t::ip6Addr
- ipIfUniqueId_t ip6IfAddrData_t::ipIfId
- uint32_t ip6IfAddrData_t::creationTime
- uint32_t ip6IfAddrData_t::lifetime
- uint8_t ip6IfAddrData_t::ip6AddrTypeAndState
- uint8_t ip6IfAddrData_t::dadTransmitCounter
- uint8_t ip6IfAddrData_t::prefixLength
- uint8_t ip6IfAddrData_t::macAddrIndex

9.2 Data Structure Documentation

9.2.1 struct ip4IfStruct_t

Information about an IPv4 interface.

Data Fields

- void(* **ip4Forward**)(ipPktInfo_t *)

9.2.2 struct ip6IfStruct_t

Information about an IPv6 interface.

Data Fields

- uint32_t scope_id
- void ** ppNdCfg
- bool_t(* ip6IsAddrOnLink)(ipAddr_t *pIpDestAddr, struct ipIfStruct_tag *instanceId)
- bool_t(* ip6ResolveUnicastAddr)(ipPktInfo_t *pIpDestAddr)
- void(* ip6UpperMgtLayerCb)(ipPktInfo_t *pIpDestAddr)
- uint32_t(* ip6McastForward)(ipPktInfo_t *, uint8_t, ipAddr_t *)
- ipAddr_t *(* ip6UnicastForward)(ipPktInfo_t *, uint8_t)

9.2.3 struct mediaIfStruct_t

Information about a media interface.

Data Fields

- uint32_t(* ifOpen)(struct ipIfStruct_tag *)
- uint32_t(* ifClose)(struct ipIfStruct_tag *)
- uint32_t(* ifSend4)(ipPktInfo_t *)
- uint32_t(* ifSendArp)(ipPktInfo_t *, llAddr_t *)
- uint32_t(* ifSend6)(ipPktInfo_t *)
- uint32_t(* ifGetIID)(struct ipIfStruct_tag *, llAddr_t *, ipAddr_t *)
- uint32_t(* ifJoin)(struct ipIfStruct_tag *, ipAddr_t *, uint16_t)
- uint32_t(* ifLeave)(struct ipIfStruct_tag *, ipAddr_t *, uint16_t)

9.2.4 struct ipIfStruct_t

Information about a network interface (IPv4 or IPv6)

Data Fields

void *	ifDriverHandle	Handle for media link layer module.
mediaIfStruct_t *	ifFunctions	Pointer to media interface functions.

uint16_t	ifMtu	Interface maximum transmission unit.
uint8_t	ipVersion4	If ipVersion4 == 1->IPv4 is enabled on this interface.
uint8_t	ipVersion6	If ipVersion6 == 1->IPv6 is enabled on this interface.
llAddr_t	ifDevAddr↩ Tbl[IP_IF_M↩ AC_ADDR_↩ NB]	Media link layer address.
ipIfUniqueId↩ _t	ifUniqueId	Interface unique ID.
uint8_t	ifMetric	Interface metric.

9.2.5 struct ip4IfAddrData_t

Information about the addressing of an IPv4 interface.

Data Fields

ipIfUniqueId↩ _t	ipIfId	Interface ID of the interface this IP4 address is binded to.
uint32_t	ip4Addr	IPv4 address in host byte order.
uint32_t	ip4SubnetMask	IPv4 address subnet mask in host byte order.
uint32_t	ip4DefaultGw	IPv4 default gateway for the interface in host byte order.

9.2.6 struct ip6IfAddrData_t

Information about the addressing of an IPv6 interface.

Data Fields

ipAddr_t	ip6Addr	IPv6 address.
ipIfUniqueId↩ _t	ipIfId	Interface ID of the interface this IP6 address is binded to.
uint32_t	creationTime	Time of entry creation (in seconds)
uint32_t	lifetime	Address lifetime expire timestamp (in seconds). 0xFFFFFFFF= Infinite Lifetime
uint8_t	ip6AddrType↩ AndState	Address type (4 bits) and current state (4 bits) .
uint8_t	dadTransmit↩ Counter	Counter used by DAD. Equals to the number of NS transmits till DAD is finished

uint8_t	prefixLength	The number of leading bits in the Prefix that are valid. -Not used, maybe used for routing
uint8_t	macAddrIndex	Index in the interface MAC address table of the MAC address this IP6 address is assigned to.

9.3 Typedef Documentation

9.3.1 typedef ipIfStruct_t* ifHandle_t

Typedef for interface handler.

9.3.2 typedef void(* ip6IfSelThreadMLSrcAddr6_t) (ipAddr_t *pDestAddr, ipAddr_t **pBestSourceAddr)

Typedef needed by the Thread Stack to select the source address when the destination address is a ML16 or ML64.

9.4 Enumeration Type Documentation

9.4.1 enum ip6AddrType_t

IPv6 address types.

Enumerator

ip6AddrTypeManual_c Manual Address.
ip6AddrTypeAutoconfigurable_c Autoconfigurable address - default MAC address.
ip6AddrTypeAutoconfigurableMac2_c Autoconfigurable address - second MAC address.

9.5 Function Documentation

9.5.1 void IP_IF_Init (void)

Init the global interface table.

9.5.2 uint32_t IP_IF_Add (ipIfUniqueId_t *ifId*, void * *driverHandle*, mediaIfStruct_t * *pIfStruct*, uint16_t *ipVersEnabled*)

Adds a new interface to the global interface table.

Parameters

in	*ifId*	Interface unique ID
in	*driverHandle*	Pointer to the packet driver handle (can be NULL)
in	*pIfStruct*	Call table for the interface
in	*ipVersEnabled*	The IP version that wants to be enabled on this interface (gIpProtv4_c, gIpProtv6_c or gIpProtv4_c | gIpProtv6_c)

Returns

uint32_t Result of the operation

9.5.3 ifHandle_t IP_IF_GetIfHandle (ipIfUniqueId_t *ifId*)

Returns pointer interface handle structure identified by unique ID.

Parameters

in	*ifId*	Interface unique ID

Returns

ifHandle_t interface handle

9.5.4 int32_t IP_IF_GetIfIndex (ipIfUniqueId_t *ipIfId*)

Returns the index (from zero) in the interface table of the provided interface.

Parameters

in	*ipIfId*	IP interface identifier

Returns

int32_t Interface index or -1 in case of error

9.5.5 bool_t IP_IF_IsMyAddr (ipIfUniqueId_t *ipIfId,* ipAddr_t * *pIpAddr*)

Checks if an unicast address is attached/bound to the interface.

Parameters

in	*ipIfId*	IP interface identifier
in	*pIpAddr*	Pointer to IP address

Returns

bool_t TRUE if the address is attached/bound, FALSE otherwise

9.5.6 void IP_IF_Join (ipIfUniqueId_t *ipIfId,* ipAddr_t * *groupIp*)

Adds a multicast group into the physical interface.

Parameters

in	*ipIfId*	IP interface identifier
in	*groupIp*	Pointer to the IP multicast address to join

9.5.7 void IP_IF_Leave (ipIfUniqueId_t *ipIfId,* ipAddr_t * *groupIp*)

Removes a multicast group from the physical interface.

Parameters

in	*ipIfId*	IP interface identifier
in	*groupIp*	Pointer to the IP multicast group address to leave

9.5.8 ipIfUniqueId_t IP_IF_GetIfIdByIndex (uint32_t *ifIndex*)

Returns the interface unique ID according to its index (from zero).

Parameters

in	*ifIndex*	The interface Index

Returns

ipIfUniqueId_t Interface unique ID

9.5.9 ifHandle_t IP_IF_GetIfByIndex (uint32_t *ifIndex*)

Returns double pointer to ifNumber interface according to its index (from zero).

Parameters

in	*ifIndex*	The interface Index

Returns

ifHandle_t It returns NULL if there is no interface with the ifNumber index

9.5.10 ifHandle_t IP_IF_GetIfByAddr (ipAddr_t * *pIpAddr*)

This function returns double pointer to interface which has the provided address.

Parameters

in	*pIpAddr*	Pointer to the IP address

Returns

ifHandle_t * It returns NULL if there is no interface with the address

9.6 Variable Documentation

9.6.1 uint32_t ip6IfStruct_t::scope_id

The scope ID of the interface, useful in link-local communication.

9.6.2 void** ip6IfStruct_t::ppNdCfg

ND configuration.

9.6.3 bool_t(* ip6IfStruct_t::ip6IsAddrOnLink) (ipAddr_t *pIpDestAddr, struct ipIfStruct_tag *instanceId)

Detects if pIpDestAddr is On-link.

9.6.4 bool_t(* ip6IfStruct_t::ip6ResolveUnicastAddr) (ipPktInfo_t *pIpDestAddr)

Selects the unicast address needed to reach pIpDestAddr.

9.6.5 void(* ip6IfStruct_t::ip6UpperMgtLayerCb) (ipPktInfo_t *pIpDestAddr)

Callback that allows management layer inspection on received packets.

9.6.6 uint32_t(∗ ip6IfStruct_t::ip6McastForward) (ipPktInfo_t ∗, uint8_t, ipAddr_t ∗)

IPv6 multicast forwarding callback.

9.6.7 ipAddr_t∗(∗ ip6IfStruct_t::ip6UnicastForward) (ipPktInfo_t ∗, uint8_t)

IPv6 unicast forwarding callback.

9.6.8 uint32_t(∗ mediaIfStruct_t::ifOpen) (struct ipIfStruct_tag ∗)

Open interface function pointer.

9.6.9 uint32_t(∗ mediaIfStruct_t::ifClose) (struct ipIfStruct_tag ∗)

Close interface function pointer.

9.6.10 uint32_t(∗ mediaIfStruct_t::ifSend4) (ipPktInfo_t ∗)

Send IPv4 packet.

9.6.11 uint32_t(∗ mediaIfStruct_t::ifSendArp) (ipPktInfo_t ∗, llAddr_t ∗)

Send IPv4 ARP packet.

9.6.12 uint32_t(∗ mediaIfStruct_t::ifSend6) (ipPktInfo_t ∗)

Send IPv6 packet.

9.6.13 uint32_t(∗ mediaIfStruct_t::ifGetIID) (struct ipIfStruct_tag ∗, llAddr_t ∗, ipAddr_t ∗)

Get the interface identifier.

9.6.14 uint32_t(∗ mediaIfStruct_t::ifJoin) (struct ipIfStruct_tag ∗, ipAddr_t ∗, uint16_t)

Join a group on the physical interface.

9.6.15 uint32_t(∗ mediaIfStruct_t::ifLeave) (struct ipIfStruct_tag ∗, ipAddr_t ∗, uint16_t)

Leave a group on the physical interface.

9.6.16 void∗ ipIfStruct_t::ifDriverHandle

Handle for media link layer module.

9.6.17 mediaIfStruct_t∗ ipIfStruct_t::ifFunctions

Pointer to media interface functions.

9.6.18 uint16_t ipIfStruct_t::ifMtu

Interface maximum transmission unit.

9.6.19 uint8_t ipIfStruct_t::ipVersion4

If ipVersion4 == 1->IPv4 is enabled on this interface.

9.6.20 uint8_t ipIfStruct_t::ipVersion6

If ipVersion6 == 1->IPv6 is enabled on this interface.

9.6.21 llAddr_t ipIfStruct_t::ifDevAddrTbl[IP_IF_MAC_ADDR_NB]

Media link layer address.

9.6.22 ipIfUniqueId_t ipIfStruct_t::ifUniqueId

Interface unique ID.

9.6.23 uint8_t ipIfStruct_t::ifMetric

Interface metric.

9.6.24 ipIfUniqueId_t ip4IfAddrData_t::ipIfId

Interface ID of the interface this IP4 address is binded to.

9.6.25 uint32_t ip4IfAddrData_t::ip4Addr

IPv4 address in host byte order.

9.6.26 uint32_t ip4IfAddrData_t::ip4SubnetMask

IPv4 address subnet mask in host byte order.

9.6.27 uint32_t ip4IfAddrData_t::ip4DefaultGw

IPv4 default gateway for the interface in host byte order.

9.6.28 ipAddr_t ip6IfAddrData_t::ip6Addr

IPv6 address.

9.6.29 ipIfUniqueId_t ip6IfAddrData_t::ipIfId

Interface ID of the interface this IP6 address is binded to.

9.6.30 uint32_t ip6IfAddrData_t::creationTime

Time of entry creation (in seconds)

9.6.31 uint32_t ip6IfAddrData_t::lifetime

Address lifetime expire timestamp (in seconds).

0xFFFFFFFF= Infinite Lifetime

9.6.32 uint8_t ip6IfAddrData_t::ip6AddrTypeAndState

Address type (4 bits) and current state (4 bits) .

9.6.33 uint8_t ip6IfAddrData_t::dadTransmitCounter

Counter used by DAD.

Equals to the number of NS transmits till DAD is finished

9.6.34 uint8_t ip6IfAddrData_t::prefixLength

The number of leading bits in the Prefix that are valid.

-Not used, maybe used for routing

9.6.35 uint8_t ip6IfAddrData_t::macAddrIndex

Index in the interface MAC address table of the MAC address this IP6 address is assigned to.

Chapter 10
Thread Network Utilities Interface

10.1 Overview

Files

- file network_utils.h

Data Structures

- union uuint16_t
- union uuint32_t
- union uuint64_t
- union ipAddr_t
- struct nwkBuffer_t
- struct llAddr_t
- struct ip6Header_t
- struct ipPktOptions_t
- struct recvOptions_t
- struct ipPktInfo_t
- union ipPktInfo_t.prot
- struct nwkMsg_t
- struct taskMsgQueue_t
- struct lut8_t
- struct nwkStats_t
- struct ipPrefix_t
- struct pbkdf2Params_t

Macros

- #define THR_ALL_FFs64
- #define THR_ALL_FFs32
- #define THR_ALL_FFs16
- #define THR_ALL_FFs8
- #define INET_ADDRSTRLEN
- #define INET6_ADDRSTRLEN
- #define INET6_IID_LEN
- #define IP6_MINIMUM_MTU
- #define IP6_PSEUDO_HDR_SIZE
- #define IP4_PSEUDO_HDR_SIZE
- #define IP4_ADDR_ANY
- #define IP4_ADDR_LOOPBACK
- #define IP4_ADDR_ALLHOSTS_GROUP
- #define IP4_ADDR_ALLROUTERS_GROUP
- #define IP4_ADDR_RIP_GROUP
- #define IP4_ADDR_NTP_GROUP
- #define IP4_ADDR_IGMP_GROUP

- #define IP4_ADDR_BROADCAST
- #define INADDR_ANY_INIT
- #define INADDR_BCAST_INIT
- #define IP4_ZERONET(a)
- #define IP4_LOOPBACK(a)
- #define IP4_MULTICAST(a)
- #define IP4_LOCAL_MULTICAST(a)
- #define IP4_EXPERIMENTAL(a)
- #define IP4_CLASS_A(a)
- #define IP4_CLASS_A_MASK
- #define IP4_CLASS_B(a)
- #define IP4_CLASS_B_MASK
- #define IP4_CLASS_C(a)
- #define IP4_CLASS_C_MASK
- #define IN6ADDR_ANY_INIT
- #define IN6ADDR_LOOPBACK_INIT
- #define IN6ADDR_NODELOCAL_ALLNODES_INIT
- #define IN6ADDR_INTFACELOCAL_ALLNODES_INIT
- #define IN6ADDR_LINKLOCAL_ALLNODES_INIT
- #define IN6ADDR_LINKLOCAL_ALLROUTERS_INIT
- #define IN6ADDR_LINKLOCAL_ALLV2ROUTERS_INIT
- #define IN6ADDR_LINKLOCAL_ALL_DHCP_ROUTERS_AND_RELAY_AGENTS
- #define IN6ADDR_REALMLOCAL_ALL_DHCP_LEASEQUERY_SERVERS
- #define IN6ADDR_REALMLOCAL_MCAST_3EAD
- #define IN6ADDR_SITELOCAL_ALLDHCPSERVERS
- #define IN6ADDR_REALMLOCAL_ALLNODES_INIT
- #define IN6ADDR_REALMLOCAL_ALLROUTERS_INIT
- #define IN6ADDR_SITELOCAL_ALLNODES_INIT
- #define IN6ADDR_SITELOCAL_ALLROUTERS_INIT
- #define IN6ADDR_LINK_LOCAL_PREFIX_INIT
- #define IN6ADDR_ALL_FFs
- #define IP_AddrCopy(dst, src)
- #define IP4_AddrToUint32(addr)
- #define IP_IsAddrEqual(addr1, addr2)
- #define IP6_IsUnspecifiedAddr(addr)
- #define IP6_IsLinkLocalAddr(addr)
- #define IP6_IsSiteLocalAddr(addr)
- #define IP6_IsUniqueLocalAddr(addr)
- #define IP6_IsGlobalAddr(addr)
- #define IP6_IsMulticastAddr(addr)
- #define IP6_IsAnycastAddr(addr)
- #define IP6_IsLoopbackAddr(addr)
- #define IP6_IsLocalMulticastAllNodes(addr)
- #define IP6_IsLocalMulticastAllRouters(addr)
- #define IP6_IsMeshMulticastAllNodes(addr)
- #define IP6_IsAddrEui64(addr)
- #define IP_ADDR(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14, a15, a16)
- #define IPV4_Mask32_g
- #define IP_IsAddrIPv4(addr)
- #define IP4_IsUnspecifiedAddr(addr)
- #define IP_IsAddrIPv6(addr)
- #define NWKU_AppendNwkBuffer(dst, src)
- #define NWKU_IsLlAddrValid(llAddr)
- #define NWKU_GetLastArrayIndex(arraySize)
- #define htona24(p, x)
- #define ntoha24(p)

- #define htona48(p, x)
- #define ntoha48(p)
- #define ntohs(val)
- #define htons(val)
- #define ntohl(val)
- #define htonl(val)
- #define ntohll(val)
- #define htonll(val)
- #define ntohas(p)
- #define htonas(p, x)
- #define ntohal(p)
- #define htonal(p, x)
- #define ntohall(p)
- #define htonall(p, x)
- #define AF_UNSPEC
- #define AF_INET
- #define AF_INET6
- #define DEFAULT_LLADDR_IDX
- #define MIN(a, b)
- #define NWKU_GENERIC_MSG_EVENT
- #define NWKU_MEM_BufferAlloc(a)
- #define NWKU_MEM_BufferAllocForever(a)

Typedefs

- typedef void(∗ nwkMsgHandler) (void ∗)
- typedef void(∗ tspDataIndCb_t) (uint8_t tspConnIndex)

Enumerations

- enum llAddrSize_t {
  gLlayerAddrNoAddr_c,
  gLlayerAddrReserved_c,
  gLlayerAddrEui16_c,
  gLlayerAddrEui48_c,
  gLlayerAddrEui64_c }
- enum ipIfUniqueId_t {
  gIpIfSlp0_c,
  gIpIfSlp1_c,
  gIpIfEth0_c,
  gIpIfEth1_c,
  gIpIfWifi0_c,
  gIpIfWifi1_c,
  gIpIfUsbRndis_c,
  gIpIfSerialTun_c,
  gIpIfBle0_c,
  gIpIfBle1_c,
  gIpIfUndef_c }
- enum nwkStatus_t {

gNwkStatusSuccess_c,
gNwkStatusMemAllocErr_c,
gNwkStatusNotAllowed_c,
gNwkStatusFail_c }

Functions

- bool_t NWKU_SendMsg (nwkMsgHandler pFunc, void *pPload, taskMsgQueue_t *msgQueue)
- void NWKU_RecvMsg (taskMsgQueue_t *pMsgQueue)
- bool_t NWKU_MsgHandler (taskMsgQueue_t *pMsgQueue)
- ipAddr_t * NWKU_CreateIpAddr (void)
- void NWKU_ConvertIp4Addr (uint32_t ip4Addr, ipAddr_t *pOutIpAddr)
- bool_t IP6_IsRealmLocalAddr (ipAddr_t *pIpAddr)
- ipPktInfo_t * NWKU_CreateIpPktInfo (void)
- void NWKU_FreeIpPktInfo (ipPktInfo_t **pIpPktInfo)
- nwkBuffer_t * NWKU_CreateNwkBuffer (uint32_t dataSize)
- void NWKU_FreeAllNwkBuffers (nwkBuffer_t **pNwkBufferStart)
- void NWKU_FreeNwkBufferElem (nwkBuffer_t **pNwkBufferStart, nwkBuffer_t *pElem)
- uint32_t NWKU_NwkBufferTotalSize (nwkBuffer_t *pNwkBufferStart)
- void NWKU_MemCopyFromNwkBuffer (nwkBuffer_t **pNwkBuffer, uint8_t **pSrcPtr, uint8_t *pDstPtr, uint32_t size)
- void NWKU_NwkBufferAddOffset (nwkBuffer_t **pNwkBuffer, uint8_t **pSrcPtr, uint32_t size)
- uint32_t NWKU_NwkBufferNumber (nwkBuffer_t *pNwkBufferStart)
- uint8_t * NWKU_NwkBufferToRegularBuffer (nwkBuffer_t *pNwkBufferStart, uint8_t *pRegularBuffer)
- void NWKU_CreatePseudoHeader4 (nwkBuffer_t *pNwkBuff, ipAddr_t *pSrcIp, ipAddr_t *pDstIp, uint32_t length, uint8_t nextHeader)
- void NWKU_CreatePseudoHeader6 (nwkBuffer_t *pNwkBuff, ipAddr_t *pSrcIp, ipAddr_t *pDstIp, uint32_t length, uint8_t nextHeader)
- uint16_t NWKU_CalculateChecksum (nwkBuffer_t *pStart)
- bool_t NWKU_CmpAddrPrefix6 (uint8_t *addr1, uint8_t *addr2, uint32_t prefixLen)
- bool_t NWKU_MemCmpToVal (uint8_t *pAddr, uint8_t val, uint32_t len)
- bool_t NWKU_BitCmp (uint8_t *pStr1, uint8_t *pStr2, uint8_t startBit, uint8_t stopBit)
- bool_t NWKU_IsLLAddrEqual (uint8_t *pFirstLlAddr, uint32_t firstLlAddrSize, uint8_t *pSecondLlAddr, uint32_t secondLlAddrSize)
- uint32_t NWKU_GetCommonPrefixLen6 (ipAddr_t *addr1, ipAddr_t *addr2)
- uint64_t NWKU_TransformArrayToValue (uint8_t *pArray, uint32_t nbOfBytes)
- void NWKU_TransformValueToArray (uint64_t value, uint8_t *pArray, uint32_t nbOfBytes)
- uint16_t NWKU_Revert16 (uint16_t value)
- uint32_t NWKU_Revert32 (uint32_t value)
- uint64_t NWKU_Revert64 (uint64_t value)
- uint16_t NWKU_TransformArrayToUint16 (uint8_t *pArray)
- uint32_t NWKU_TransformArrayToUint32 (uint8_t *pArray)
- uint64_t NWKU_TransformArrayToUint64 (uint8_t *pArray)
- void NWKU_TransformUint16ToArray (uint8_t *pArray, uint16_t value)
- void NWKU_TransformUint32ToArray (uint8_t *pArray, uint32_t value)
- void NWKU_TransformUint64ToArray (uint8_t *pArray, uint64_t value)
- bool_t NWKU_GetLut8 (lut8_t *pLutTable, uint8_t lutTableSize, uint8_t type, uint8_t *pEntryIndex)
- int32_t NWKU_atoi (char *pStr)
- int64_t NWKU_atol (char *pStr)
- void NWKU_PrintDec (uint64_t value, uint8_t *pString, uint32_t nbPrintDigits, bool_t bLeadingZeros)

- int32_t pton (uint8_t af, char *pTxt, ipAddr_t *pIpAddr)
- char * ntop (uint8_t af, ipAddr_t *pIpAddr, char *pStr, uint32_t strLen)
- bool_t ptoll (uint8_t *pIn, uint32_t len, llAddr_t *pLlAddr)
- uint32_t NWKU_AsciiToHex (uint8_t *pString, uint32_t strLen)
- uint32_t NWKU_AsciiToDec (uint8_t *pString, uint32_t strLen)
- uint8_t NWKU_ByteToDec (uint8_t byte)
- uint8_t NWKU_NibToAscii (int8_t nib, bool_t useUpperCase)
- void NWKU_HexToAscii (uint8_t *pInputBuff, uint32_t inputBuffLen, uint8_t *pOutputBuffer, uint32_t outputBuffLen, bool_t useUpperCase)
- uint32_t NWKU_TmrRtcGetElapsedTimeInSeconds (uint32_t timestamp)
- bool_t NWKU_IsNUmber (char *pString)
- uint32_t NWKU_GetRandomNoFromInterval (uint32_t startInterval, uint32_t endInterval)
- void NWKU_IncrementIp6Addr (ipAddr_t *pIpAddr)
- uint32_t NWKU_RightRotate (uint32_t val, uint8_t amount)
- void NWKU_GetIIDFromLLADDR (llAddr_t *llAddr, uint16_t panId, uint8_t *pIID)
- void NWKU_GetLLAddrFromIID (uint8_t *pIID, llAddr_t *pLlAddr)
- bool_t NWKU_IsIPAddrBasedOnShort (ipAddr_t *pIpAddr)
- bool_t NWKU_GetBit (uint32_t bitNr, uint8_t *pArray)
- void NWKU_SetBit (uint32_t bitNr, uint8_t *pArray)
- void NWKU_ClearBit (uint32_t bitNr, uint8_t *pArray)
- uint32_t NWKU_GetFirstBitValueInRange (uint8_t *pArray, uint32_t lowBitNr, uint32_t highBitNr, bool_t bitValue)
- uint32_t NWKU_GetFirstBitValue (uint8_t *pArray, uint32_t arrayBytes, bool_t bitValue)
- uint32_t NWKU_GetNumOfBits (uint8_t *pArray, uint32_t arrayBytes, bool_t bitValue)
- uint32_t NWKU_ReverseBits (uint32_t num)
- uint32_t NWKU_AddTblEntry (uint32_t entry, uint32_t *pTable, uint32_t tableSize)
- uint32_t NWKU_GetTblEntry (uint32_t index, uint32_t *pTable, uint32_t tableSize)
- void NWKU_SwapArrayBytes (uint8_t *pByte, uint8_t numOfBytes)
- void NWKU_GenRand (uint8_t *pRand, uint8_t randLen)
- uint32_t NWKU_GetTlvLen (uint8_t type, uint8_t *pStart, uint32_t len)
- uint8_t * NWKU_GetTlvValue (uint8_t type, uint8_t *pStart, uint32_t len, uint8_t *pOut)
- uint8_t * NWKU_GetTlv (uint8_t type, uint8_t *pStart, uint32_t len, uint8_t *pOut, uint32_t *pOutLen)
- bool_t NWKU_Pbkdf2 (pbkdf2Params_t *pInput, uint8_t *pOut, uint32_t outLen)
- uint64_t NWKU_GetTimestampMs (void)
- int8_t NWKU_isArrayGreater (const uint8_t *a, const uint8_t *b, uint8_t length)

Variables

- uint16_t uuint16_t::u16
- uint8_t uuint16_t::u8 [2]
- uint32_t uuint32_t::u32
- uint16_t uuint32_t::u16 [2]
- uint8_t uuint32_t::u8 [4]
- uint64_t uuint64_t::u64
- uint32_t uuint64_t::u32 [2]
- uint16_t uuint64_t::u16 [4]
- uint8_t uuint64_t::u8 [8]
- uint8_t ipAddr_t::addr8 [16]
- uint16_t ipAddr_t::addr16 [8]
- uint32_t ipAddr_t::addr32 [4]
- uint64_t ipAddr_t::addr64 [2]
- struct nwkBuffer_tag * nwkBuffer_t::next
- uint8_t * nwkBuffer_t::pData
- uint32_t nwkBuffer_t::size

- uint8_t nwkBuffer_t::freeBuffer
- uint8_t llAddr_t::eui [8]
- llAddrSize_t llAddr_t::addrSize
- uint8_t ip6Header_t::versionTraficClass
- uint8_t ip6Header_t::trafficClassFlowLabel
- uint8_t ip6Header_t::flowLabel [2]
- uint8_t ip6Header_t::payloadLength [2]
- uint8_t ip6Header_t::nextHeader
- uint8_t ip6Header_t::hopLimit
- uint8_t ip6Header_t::srcAddr [16]
- uint8_t ip6Header_t::dstAddr [16]
- void * ipPktOptions_t::ifHandle
- nwkBuffer_t * ipPktOptions_t::ipExtensionHeaderBuffer
- void * ipPktOptions_t::ipReassemblyOptions
- llAddr_t ipPktOptions_t::srcLlInfo
- uint8_t ipPktOptions_t::ipHdrOffset
- uint8_t ipPktOptions_t::hopLimit
- uint8_t ipPktOptions_t::security
- uint8_t ipPktOptions_t::lqi
- uint8_t ipPktOptions_t::qos
- uint8_t ipPktOptions_t::isRelay
- uint8_t ipPktOptions_t::macSecKeyIdMode
- uint8_t ipPktOptions_t::channel
- uint16_t ipPktOptions_t::destPanId
- uint16_t ipPktOptions_t::srcPanId
- ipIfUniqueId_t recvOptions_t::ipIfId
- uint8_t recvOptions_t::hopLimit
- uint8_t recvOptions_t::security
- uint8_t recvOptions_t::lqi
- uint8_t recvOptions_t::isRelay
- uint8_t recvOptions_t::channel
- uint8_t recvOptions_t::macSecKeyIdMode
- uint16_t recvOptions_t::macSrcPanId
- nwkBuffer_t * ipPktInfo_t::pNwkBuff
- ipAddr_t * ipPktInfo_t::pIpSrcAddr
- ipAddr_t * ipPktInfo_t::pIpDstAddr
- uint8_t * ipPktInfo_t::pNextProt
- ipAddr_t ipPktInfo_t::ipSrcAddr
- ipAddr_t ipPktInfo_t::ipDstAddr
- uint32_t ipPktInfo_t::nextProtLen
- uint32_t ipPktInfo_t::protocolType
- union {
  uint32_t nextProtLen
  uint32_t protocolType
  } ipPktInfo_t::prot

- uint16_t ipPktInfo_t::srcPort
- uint16_t ipPktInfo_t::dstPort
- ipPktOptions_t ipPktInfo_t::ipPktOptions
- nwkMsgHandler nwkMsg_t::pFunc
- void * nwkMsg_t::pPload
- msgQueue_t taskMsgQueue_t::msgQueue
- osaTaskId_t taskMsgQueue_t::taskId
- osaEventId_t taskMsgQueue_t::taskEventId
- uint8_t lut8_t::type

- uint8_t lut8_t::idx
- uint8_t nwkStats_t::ipktUsed
- uint8_t nwkStats_t::ipktMax
- uint8_t nwkStats_t::nwkBuffUsed
- uint8_t nwkStats_t::nwkBuffMax
- uint8_t ipPrefix_t::prefixLen
- uint8_t ipPrefix_t::aPrefix [ ]
- uint8_t * pbkdf2Params_t::pPass
- uint32_t pbkdf2Params_t::passLen
- uint8_t * pbkdf2Params_t::pSalt
- uint32_t pbkdf2Params_t::saltLen
- uint32_t pbkdf2Params_t::rounds
- const ipAddr_t **inaddr_any**
- const ipAddr_t **inaddr_bcast**
- const ipAddr_t **in6addr_any**
- const ipAddr_t **in6addr_loopback**
- const ipAddr_t **in6addr_nodelocal_allnodes**
- const ipAddr_t **in6addr_linklocal_allnodes**
- const ipAddr_t **in6addr_linklocal_allrouters**
- const ipAddr_t **in6addr_linklocal_allv2routers**
- const ipAddr_t **in6addr_sitelocal_alldhcpservers**
- const ipAddr_t **in6addr_realmlocal_allnodes**
- const ipAddr_t **in6addr_realmlocal_allrouters**
- const ipAddr_t **in6addr_realmlocal_allleasequeryservers**
- const ipAddr_t **in6addr_realmlocal_mcast_3ead**
- const ipAddr_t **in6addr_sitelocal_allnodes**
- const ipAddr_t **in6addr_sitelocal_allrouters**
- const ipAddr_t **in6addr_link_local_prefix**
- ipAddr_t **in6addr_linklocal_allthreadnodes**
- ipAddr_t **in6addr_realmlocal_allthreadnodes**
- ipAddr_t **in6addr_realmlocal_threadleaderanycast**
- const uint8_t **gNwkPoolId**

10.2 Data Structure Documentation

10.2.1 union uuint16_t

Generic structure for holding uint16 values.

Data Fields

uint16_t	u16	16bit variable
uint8_t	u8[2]	8bit array

10.2.2 union uuint32_t

Generic structure for holding uint32 values.

Data Fields

uint32_t	u32	32bit variable
uint16_t	u16[2]	16bit array
uint8_t	u8[4]	8bit array

10.2.3 union uuint64_t

Generic structure for holding uint64 values.

Data Fields

uint64_t	u64	64bit variable
uint32_t	u32[2]	32bit array
uint16_t	u16[4]	16bit array
uint8_t	u8[8]	8bit array

10.2.4 union ipAddr_t

Generic structure for holding IP address information.

Data Fields

uint8_t	addr8[16]	8bit array
uint16_t	addr16[8]	16bit array
uint32_t	addr32[4]	32bit array
uint64_t	addr64[2]	64bit array

10.2.5 struct nwkBuffer_t

Generic structure for holding buffer information.

Data Fields

struct nwkBuffer_tag *	next	Pointer to next buffer.
uint8_t *	pData	Pointer to data.
uint32_t	size	Size of data.
uint8_t	freeBuffer	Flag used to notify buffer clearance.

10.2.6 struct llAddr_t

Generic structure for link layer address.

Data Fields

uint8_t	eui[8]	Destination address: short/extended.
llAddrSize_t	addrSize	Destination address type: short/extended.

10.2.7 struct ip6Header_t

Generic structure for IPv6 header.

Data Fields

uint8_t	versionTrafic↩ Class	Version Traffic Class.
uint8_t	trafficClass↩ FlowLabel	Traffic Class Flow label.
uint8_t	flowLabel[2]	Flow label.
uint8_t	payload↩ Length[2]	Payload length.
uint8_t	nextHeader	Next header.
uint8_t	hopLimit	Hop limit.
uint8_t	srcAddr[16]	Source Address.
uint8_t	dstAddr[16]	Destination Address.

10.2.8 struct ipPktOptions_t

Generic structure for IP packet options.

Data Fields

void *	ifHandle	Pointer to interface handler.
nwkBuffer_t *	ipExtension↩ HeaderBuffer	Pointer to extended options buffer.
void *	ip↩ Reassembly↩ Options	Pointer to IP reassembly structure.
llAddr_t	srcLlInfo	Source Link Layer information.
uint8_t	ipHdrOffset	Offset from beginning of RX data where IP HDR is found.
uint8_t	hopLimit	Hop limit.
uint8_t	security	Security option.
uint8_t	lqi	Packet LQI.

uint8_t	qos	Packet Quality of Service.
uint8_t	isRelay	Flag to specify if packet is relay.
uint8_t	macSecKeyId↩ Mode	MacSec Key ID Mode.
uint8_t	channel	Packet Channel.
uint16_t	destPanId	Destination PAN ID.
uint16_t	srcPanId	Source PAN ID.

10.2.9 struct recvOptions_t

Received packet options structure.

Data Fields

ipIfUniqueId↩ _t	ipIfId	ID of the interface.
uint8_t	hopLimit	Hop limit.
uint8_t	security	Security option.
uint8_t	lqi	Packet LQI.
uint8_t	isRelay	Flag to specify if packet is relay.
uint8_t	channel	Packet Channel.
uint8_t	macSecKeyId↩ Mode	MacSec Key ID Mode.
uint16_t	macSrcPanId	MAC Source PAN ID.

10.2.10 struct ipPktInfo_t

Data Fields

nwkBuffer_t *	pNwkBuff	Pointer to network buffer.
ipAddr_t *	pIpSrcAddr	Pointer to source IP address.
ipAddr_t *	pIpDstAddr	Pointer to destination IP address.
uint8_t *	pNextProt	Pointer to the next protocol in pNwkBuff->pData. Do not free this one!
ipAddr_t	ipSrcAddr	Source IP address.
ipAddr_t	ipDstAddr	Destination IP address.
union ipPktInfo_t	prot	Protocol information.

uint16_t	srcPort	Source port.
uint16_t	dstPort	Destination port.
ipPktOptions_t	ipPktOptions	IP packet options.

10.2.11 union ipPktInfo_t.prot

Protocol information.

Data Fields

uint32_t	nextProtLen	Size of the data of next protocol in pNwkBuff->pData.
uint32_t	protocolType	Protocol type.

10.2.12 struct nwkMsg_t

Generic structure for network message.

Data Fields

nwkMsgHandler	pFunc	Pointer to packet handler.
void *	pPload	Pointer to handler payload.

10.2.13 struct taskMsgQueue_t

Task Message Queue structure.

Data Fields

msgQueue_t	msgQueue	Pointer to task message queue.
osaTaskId_t	taskId	Pointer to task ID.
osaEventId_t	taskEventId	Pointer to task event ID.

10.2.14 struct lut8_t

Lookup tables with 8 bits elements.

Data Fields

uint8_t	type	Type.
uint8_t	idx	Index.

10.2.15 struct nwkStats_t

Network statistics, for debug.

Data Fields

uint8_t	ipktUsed	IP packets used.
uint8_t	ipktMax	Maximum IP packets.
uint8_t	nwkBuffUsed	Network buffers used.
uint8_t	nwkBuffMax	Maximum network buffers.

10.2.16 struct ipPrefix_t

Structure for holding IP prefix information.

Data Fields

uint8_t	prefixLen	Size of the prefix in bits.
uint8_t	aPrefix[ ]	Pointer to the start of the prefix.

10.2.17 struct pbkdf2Params_t

Structure used for pbkdf2 generation.

Data Fields

uint8_t *	pPass	Pointer to the password.
uint32_t	passLen	Length of the password.
uint8_t *	pSalt	Pointer to the salt.
uint32_t	saltLen	Length of the salt.
uint32_t	rounds	Number of rounds.

10.3 Macro Definition Documentation

10.3.1 #define THR_ALL_FFs64

Max unsigned 64bit integers value.

10.3.2 #define THR_ALL_FFs32

Max unsigned 32bit integers value.

10.3.3 #define THR_ALL_FFs16

Max unsigned 16bit integers value.

10.3.4 #define THR_ALL_FFs8

Max unsigned 8bit integers value.

10.3.5 #define INET_ADDRSTRLEN

Length for IP address string size (used to compute size used in ntop).
Value for 16 bytes strings

10.3.6 #define INET6_ADDRSTRLEN

Length for IP address string size (used to compute size used in ntop).
Value for 46 bytes strings

10.3.7 #define INET6_IID_LEN

Length for IP address string size (used to compute size used in ntop).
Value for IID strings

10.3.8 #define IP6_MINIMUM_MTU

Minimum MTU value.

10.3.9 #define IP6_PSEUDO_HDR_SIZE

IPv6 Pseudo HDR size.

10.3.10 #define IP4_PSEUDO_HDR_SIZE

IPv4 Pseudo HDR size.

10.3.11 #define IP4_ADDR_ANY

IPv4 any address.

10.3.12 #define IP4_ADDR_LOOPBACK

IPv4 loopback address.

10.3.13 #define IP4_ADDR_ALLHOSTS_GROUP

IPv4 all host group address.

10.3.14 #define IP4_ADDR_ALLROUTERS_GROUP

IPv4 all routers group address.

10.3.15 #define IP4_ADDR_RIP_GROUP

IPv4 RIP group address.

10.3.16 #define IP4_ADDR_NTP_GROUP

IPv4 NTP group address.

10.3.17 #define IP4_ADDR_IGMP_GROUP

IPv4 IGMP group address.

10.3.18 #define IP4_ADDR_BROADCAST

IPv4 all routers group address.

10.3.19 #define INADDR_ANY_INIT

IPv4 any address mapped to IPv6.

10.3.20 #define INADDR_BCAST_INIT

IPv4 broadcast address mapped to IPv6.

10.3.21 #define IP4_ZERONET(*a*)

Macro to classify IPv4 address to any.

10.3.22 #define IP4_LOOPBACK(*a*)

Macro to classify IPv4 address to loopback.

10.3.23 #define IP4_MULTICAST(*a*)

Macro to classify IPv4 address to multicast.

10.3.24 #define IP4_LOCAL_MULTICAST(*a*)

Macro to classify IPv4 address to local multicast.

10.3.25 #define IP4_EXPERIMENTAL(*a*)

Macro to classify IPv4 address to experimental.

10.3.26 #define IP4_CLASS_A(*a*)

Macro to classify IPv4 address to class A.

10.3.27 #define IP4_CLASS_A_MASK

IPv4 Class A mask.

10.3.28 #define IP4_CLASS_B(*a*)

Macro to classify IPv4 address to class B.

10.3.29 #define IP4_CLASS_B_MASK

IPv4 Class B mask.

10.3.30 #define IP4_CLASS_C(*a*)

Macro to classify IPv4 address to class C.

10.3.31 #define IP4_CLASS_C_MASK

IPv4 Class C mask.

10.3.32 #define IN6ADDR_ANY_INIT

IPV6 any address.

10.3.33 #define IN6ADDR_LOOPBACK_INIT

IPV6 loopback address.

10.3.34 #define IN6ADDR_NODELOCAL_ALLNODES_INIT

IPV6 node local all nodes address.

10.3.35 #define IN6ADDR_INTFACELOCAL_ALLNODES_INIT

IPV6 interface local all nodes address.

10.3.36 #define IN6ADDR_LINKLOCAL_ALLNODES_INIT

IPV6 link local all nodes address.

10.3.37 #define IN6ADDR_LINKLOCAL_ALLROUTERS_INIT

IPV6 link local all routers address.

10.3.38 #define IN6ADDR_LINKLOCAL_ALLV2ROUTERS_INIT

IPV6 link local all v2 routers address.

10.3.39 #define IN6ADDR_LINKLOCAL_ALL_DHCP_ROUTERS_AND_RELAY_AG↩ ENTS

IPV6 link local all DHCP routers and relay agents address.

10.3.40 #define IN6ADDR_REALMLOCAL_ALL_DHCP_LEASEQUERY_SERVERS

IPV6 realm local all DHCP lease query servers address.

10.3.41 #define IN6ADDR_REALMLOCAL_MCAST_3EAD

IPV6 realm local multicast 3ead address.

10.3.42 #define IN6ADDR_SITELOCAL_ALLDHCPSERVERS

IPV6 site local all DHCP servers address.

10.3.43 #define IN6ADDR_REALMLOCAL_ALLNODES_INIT

IPV6 realm local all nodes address.

10.3.44 #define IN6ADDR_REALMLOCAL_ALLROUTERS_INIT

IPV6 realm local all routers address.

10.3.45 #define IN6ADDR_SITELOCAL_ALLNODES_INIT

IPV6 site local all nodes address.

10.3.46 #define IN6ADDR_SITELOCAL_ALLROUTERS_INIT

IPV6 site local all routers address.

10.3.47 #define IN6ADDR_LINK_LOCAL_PREFIX_INIT

IPV6 link local prefix address.

10.3.48 #define IN6ADDR_ALL_FFs

IPV6 all FFs address.

10.3.49 #define IP_AddrCopy(*dst, src*)

Macro for IP address copy.

10.3.50 #define IP4_AddrToUint32(*addr*)

Macro for IP address conversion to uint32_t.

10.3.51 #define IP_IsAddrEqual(*addr1, addr2*)

Macro for IPV6 address comparison.

10.3.52 #define IP6_IsUnspecifiedAddr(*addr*)

Macro for unspecified IPV6 address inquiry.

10.3.53 #define IP6_IsLinkLocalAddr(*addr*)

Macro for link local IPV6 address inquiry.

10.3.54 #define IP6_IsSiteLocalAddr(*addr*)

Macro for site local IPV6 address inquiry.

10.3.55 #define IP6_IsUniqueLocalAddr(*addr*)

Macro for unique local IPV6 address inquiry.

10.3.56 #define IP6_IsGlobalAddr(*addr*)

Macro for global IPV6 address inquiry.

10.3.57 #define IP6_IsMulticastAddr(*addr*)

Macro for multicast IPV6 address inquiry.

10.3.58 #define IP6_IsAnycastAddr(*addr*)

Macro for anycast IPV6 address inquiry.

10.3.59 #define IP6_IsLoopbackAddr(*addr*)

Macro for loopback IPV6 address inquiry.

10.3.60 #define IP6_IsLocalMulticastAllNodes(*addr*)

Macro for local multicast all nodes IPV6 address inquiry.

10.3.61 #define IP6_IsLocalMulticastAllRouters(*addr*)

Macro for local multicast all routers IPV6 address inquiry.

10.3.62 #define IP6_IsMeshMulticastAllNodes(*addr*)

Macro for mesh multicast all nodes IPV6 address inquiry.

10.3.63 #define IP6_IsAddrEui64(*addr*)

Macro for EUI64 IPV6 address inquiry.

10.3.64 #define IP_ADDR(*a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14, a15, a16*)

Macro for values to IP address array transformation.

10.3.65 #define IPV4_Mask32_g

Mask for IPV4 address identification(RFC4291: 2.5.5.2)

10.3.66 #define IP_IsAddrIPv4(*addr*)

Macro for IPV4 in IPv6 address inquiry.

10.3.67 #define IP4_IsUnspecifiedAddr(*addr*)

Macro for IPV4 unspecified address inquiry.

10.3.68 #define IP_IsAddrIPv6(*addr*)

Macro for IPV6 address inquiry.

10.3.69 #define NWKU_AppendNwkBuffer(*dst, src*)

Macro for appending network buffer.

10.3.70 #define NWKU_IsLIAddrValid(*llAddr*)

Macro for link layer address validity inquiry.

10.3.71 #define NWKU_GetLastArrayIndex(*arraySize*)

Macro for retrieving the last index of an array.

10.3.72 #define htona24(*p, x*)

Macro for host variable to 24 bit network array conversion.

10.3.73 #define ntoha24(*p*)

Macro for 24 bit network array to host variable conversion.

10.3.74 #define htona48(*p, x*)

Macro for host variable to 48 bit network array conversion.

10.3.75 #define ntoha48(*p*)

Macro for 48 bit network array to host variable conversion.

10.3.76 #define ntohs(*val*)

Macro for network to host short conversion.

10.3.77 #define htons(*val*)

Macro for host short to network conversion.

10.3.78 #define ntohl(*val*)

Macro for network to host 32bit conversion.

10.3.79 #define htonl(*val*)

Macro for host 32bit to network conversion.

10.3.80 #define ntohll(*val*)

Macro for network to host 64bit conversion.

10.3.81 #define htonll(*val*)

Macro for host 64bit to network conversion.

10.3.82 #define ntohas(p)

Macro for network array to host short conversion.

10.3.83 #define htonas(p, x)

Macro for host short to network array conversion.

10.3.84 #define ntohal(p)

Macro for network array to host 32bit conversion.

10.3.85 #define htonal(p, x)

Macro for host 32bit to network array conversion.

10.3.86 #define ntohall(p)

Macro for network array to host 64bit conversion.

10.3.87 #define htonall(p, x)

Macro for host 64bit to network array conversion.

10.3.88 #define AF_UNSPEC

Unspecified sockets.

10.3.89 #define AF_INET

Internet IP Protocol.

10.3.90 #define AF_INET6

IP version 6.

10.3.91 #define DEFAULT_LLADDR_IDX

Default index for link layer address.

10.3.92 #define MIN(*a, b*)

Macro for obtaining the minimum value variable between two input variables.

10.3.93 #define NWKU_GENERIC_MSG_EVENT

Generic Message Event.

10.3.94 #define NWKU_MEM_BufferAlloc(*a*)

Macro for memory buffer allocation.

Parameters

in	*a*	Size of requested memory buffer

10.3.95 #define NWKU_MEM_BufferAllocForever(*a*)

Macro for memory buffer allocation.

The allocated memory buffer will never be freed

Parameters

in	*a*	Size of requested memory buffer

10.4 Typedef Documentation

10.4.1 typedef void(* nwkMsgHandler) (void *)

Callback function for servicing network messages.

10.4.2 typedef void(* tspDataIndCb_t) (uint8_t tspConnIndex)

Callback function for servicing transport packets.

Parameters

in	*tspConnIndex*	Connection index

10.5 Enumeration Type Documentation

10.5.1 enum llAddrSize_t

Enumeration for address size.

Enumerator

gLlayerAddrNoAddr_c No address (addressing fields omitted)
gLlayerAddrReserved_c Reserved value.
gLlayerAddrEui16_c 16-bit short Link Layer address (size 2 bytes)
gLlayerAddrEui48_c 48-bit Ethernet MAC Address (size 6 bytes)
gLlayerAddrEui64_c 64-bit extended Link Layer address (size 8 bytes)

10.5.2 enum ipIfUniqueId_t

Unique interface ID enumeration.

Enumerator

gIpIfSlp0_c SLWP0 interface.
gIpIfSlp1_c SLWP1 interface.
gIpIfEth0_c ETH0 interface.
gIpIfEth1_c ETH1 interface.
gIpIfWifi0_c WiFi0 interface.
gIpIfWifi1_c WiFi1 interface.
gIpIfUsbRndis_c RNDIS interface.
gIpIfSerialTun_c Serial TUN interface.
gIpIfBle0_c BLE0 interface.
gIpIfBle1_c BLE1 interface.
gIpIfUndef_c Undefined interface.

10.5.3 enum nwkStatus_t

Network generic status enumeration.

Enumerator

gNwkStatusSuccess_c Network Status: Success.

gNwkStatusMemAllocErr_c Network Status: Memory allocation error.
gNwkStatusNotAllowed_c Operation was not allowed.
gNwkStatusFail_c Network Status: Fail.

10.6 Function Documentation

10.6.1 bool_t NWKU_SendMsg (nwkMsgHandler *pFunc,* void * *pPload,* taskMsgQueue_t * *msgQueue*)

Network Utils module function used to send a message between two tasks.

Parameters

in	*pFunc*	Pointer to message handler function
in	*pPload*	Pointer to message data
in	*msgQueue*	Pointer to structure holding message queue and task id to send message

Returns

TRUE If the message was sent successfully
FALSE If not

10.6.2 void NWKU_RecvMsg (taskMsgQueue_t * *msgQueue*)

Network Utils module function used to receive and handle a message in a task.

Parameters

in	*pMsgQueue*	Pointer to structure holding message queue and task id to receive message

10.6.3 bool_t NWKU_MsgHandler (taskMsgQueue_t * *pMsgQueue*)

Network Utils module function used to dequeue and handle a task message.

Parameters

in	*pMsgQueue*	Pointer to structure holding message queue and task id to receive message

Returns

TRUE If there was a message in the queue
FALSE Otherwise

10.6.4 ipAddr_t * NWKU_CreateIpAddr (void)

Network Utils module function used to create an ipAddr_t structure.

Returns

Pointer to the allocated ipAddr_t structure
NULL if memory cannot be allocated

10.6.5 void NWKU_ConvertIp4Addr (uint32_t *ip4Addr,* ipAddr_t * *pOutIpAddr*)

Network Utils module function used to convert an IPv4 address in uint32_t format to an ipAddr_t type address.

Parameters

in	*ip4Addr*	IPv4 address
out	*pOutIpAddr*	Pointer to ipAddr_t to store the converted address

10.6.6 bool_t IP6_IsRealmLocalAddr (ipAddr_t * *pIpAddr*)

Network Utils module function used determine if an IPv6 address has realm local scope - valid only in the context of a THREAD stack.

Parameters

in	*pIpAddr*	IPv6 address

Returns

TRUE If address is realm local
FALSE If not or if not supported

10.6.7 ipPktInfo_t * NWKU_CreateIpPktInfo (void)

Network Utils module function used to create an ipPktInfo_t structure.

Returns

Pointer to the allocated ipPktInfo_t structure
NULL if memory cannot be allocated

10.6.8 void NWKU_FreeIpPktInfo (ipPktInfo_t ** *pIpPktInfo*)

Network Utils module function used to free one ipPktInfo_t structure.

Parameters

in	*pIpPktInfo*	Double pointer to the ipPktInfo_t structure

10.6.9 nwkBuffer_t * NWKU_CreateNwkBuffer (uint32_t *dataSize*)

Network Utils module function used to create a nwkBuffer_t structure and allocate memory for data.

Parameters

in	*dataSize*	Size of the data available in the buffer

Returns

Pointer to the allocated nwkBuffer_t structure
NULL if memory cannot be allocated

10.6.10 void NWKU_FreeAllNwkBuffers (nwkBuffer_t ** *pNwkBufferStart*)

Network Utils module function used to free all nwkBuffer_t structures(starting with pNwkBufferStart) and change the start of the list to NULL.

Parameters

in	*pNwkBuffer↩ Start*	Double pointer to the start of data buffer

10.6.11 void NWKU_FreeNwkBufferElem (nwkBuffer_t ** *pNwkBufferStart,* nwkBuffer_t * *pElem*)

Network Utils module function used to free one nwkBuffer_t element.

Parameters

in	*pNwkBuffer↩ Start*	Double pointer to the start of data buffer
in	*pElem*	Pointer to the element to be freed

10.6.12 uint32_t NWKU_NwkBufferTotalSize (nwkBuffer_t * *pNwkBufferStart*)

Network Utils module function used to calculate the total size of a nwkBuffer_t list, starting with pNwk↩ BufferStart.

Parameters

in	*pNwkBuffer↩ Start*	Pointer to the start of nwkBuffer

Returns

Size of the whole list

10.6.13 void NWKU_MemCopyFromNwkBuffer (nwkBuffer_t ** *pNwkBuffer,* uint8_t ** *pSrcPtr,* uint8_t * *pDstPtr,* uint32_t *size*)

Network Utils module function used to copy from a network fragmented buffer into a regular linear buffer.

Parameters

in,out	*pNwkBuffer*	Pointer to the start network buffer - pointer to end network buffer
in,out	*pSrcPtr*	Pointer to the source date in the start network buffer - returns last position in the end network buffer
in	*pDstPtr*	Destination pointer
in	*size*	Size to copy

10.6.14 void NWKU_NwkBufferAddOffset (nwkBuffer_t ** *pNwkBuffer,* uint8_t ** *pSrcPtr,* uint32_t *size*)

Network Utils module function used to add data into a buffer using an offset.

Parameters

in,out	*pNwkBuffer*	Pointer to the start network buffer - pointer to end network buffer
in,out	*pSrcPtr*	Pointer to the source date in the start network buffer - returns last position in the end network buffer
in	*size*	Size to copy

10.6.15 uint32_t NWKU_NwkBufferNumber (nwkBuffer_t * *pNwkBufferStart*)

Network Utils module function used to return the number of nwkBuffer_t fragments in the list

Parameters

in	*pNwkBuffer↩ Start*	Pointer to the start of data buffer

Returns

Number of nwkBuffer_t fragments in the list

10.6.16 uint8_t * NWKU_NwkBufferToRegularBuffer (nwkBuffer_t * *pNwkBufferStart*, uint8_t * *pRegularBuffer*)

Network Utils module function used to transform a network fragmented buffer into a regular linear buffer.

Parameters

in	*pNwkBuffer↩ Start*	Pointer to the start of network buffer
in	*pRegularBuffer*	Pointer to the provided Buffer, if null then allocate

Returns

Pointer to an allocated regular buffer that gets created
NULL if memory cannot be allocated

10.6.17 void NWKU_CreatePseudoHeader4 (nwkBuffer_t * *pNwkBuff*, ipAddr_t * *pSrcIp*, ipAddr_t * *pDstIp*, uint32_t *length*, uint8_t *nextHeader*)

Network Utils module function used to create the pseudoheader for IPv4 protocols

Parameters

out	*pNwkBuff*	Pointer to the nwkBuffer_t element containing the pseudoheader
in	*pSrcIp*	Pointer to the source IP address
in	*pDstIp*	Pointer to the destination IP address
in	*length*	Length of the protocol(header + data)
in	*nextHeader*	Value of the next header

10.6.18 void NWKU_CreatePseudoHeader6 (nwkBuffer_t * *pNwkBuff*, ipAddr_t * *pSrcIp*, ipAddr_t * *pDstIp*, uint32_t *length*, uint8_t *nextHeader*)

Network Utils module function used to create the pseudoheader for IPv6 protocols

Parameters

out	*pNwkBuff*	Pointer to the nwkBuffer_t element containing the pseudoheader
in	*pSrcIp*	Pointer to the source IP address
in	*pDstIp*	Pointer to the destination IP address
in	*length*	Length of the protocol(header + data)
in	*nextHeader*	Value of the next header

10.6.19 uint16_t NWKU_CalculateChecksum (nwkBuffer_t * *pStart*)

Network Utils module function used to calculate the checksum for a nwkBuffer_t list starting with pStart element

Parameters

in	*pStart*	Pointer to the start of the list

Returns

Checksum for the whole list

10.6.20 bool_t NWKU_CmpAddrPrefix6 (uint8_t * *addr1,* uint8_t * *addr2,* uint32_t *prefixLen*)

Compares first "prefixLen" bits of the ipv6 addresses.

Parameters

in	*addr1*	First prefix to compare
in	*addr2*	Second prefix to compare
in	*prefixLen*	Length in bits to compare

Returns

TRUE If match
FALSE Otherwise

10.6.21 bool_t NWKU_MemCmpToVal (uint8_t * *pAddr,* uint8_t *val,* uint32_t *len*)

Compare each octet of a given location to a value.

Parameters

in	*pAddr*	location to be compared
in	*val*	reference value
in	*len*	length of location to be compared

Returns

TRUE If match
FALSE Otherwise

10.6.22 bool_t NWKU_BitCmp (uint8_t * *pStr1,* uint8_t * *pStr2,* uint8_t *startBit,* uint8_t *stopBit*)

Compare two strings bit by bit

Parameters

in	*pStr1*	The start address of the first string to be compared
in	*pStr2*	The start address of the second string to be compared
in	*startBit*	The start bit number in the the 2 strings
in	*stopBit*	The stop bit number in the the 2 strings

Returns

TRUE If the strings match
FALSE If the strings don't match

10.6.23 bool_t NWKU_IsLLAddrEqual (uint8_t * *pFirstLlAddr,* uint32_t *firstLlAddrSize,* uint8_t * *pSecondLlAddr,* uint32_t *secondLlAddrSize*)

Compare two Link Layer addresses

Parameters

in	*pFirstLlAddr*	The start address of the first address to be compared
in	*firstLlAddrSize*	The size of the first address to be compared
in	*pSecondLlAddr*	The start address of the second address to be compared
in	*secondLl↩ AddrSize*	The size of the second address to be compared

Returns

TRUE If the Link Layer addresses are the same
FALSE If the Link Layer addresses are different

10.6.24 uint32_t NWKU_GetCommonPrefixLen6 (ipAddr_t * *addr1,* ipAddr_t * *addr2*)

The common prefix length CommonPrefixLen(A, B) of two addresses A and B is the length of the longest prefix (looking at the most significant, or leftmost, bits) that the two addresses have in common.

Parameters

in	*addr1*	First prefix to compare
in	*addr2*	Second prefix to compare

Returns

Longest prefix length in bits (0 - 128)

10.6.25 uint64_t NWKU_TransformArrayToValue (uint8_t * *pArray,* uint32_t *nbOfBytes*)

Converts an array to a numeric value.

Parameters

in	*pArray*	The start address of the array
in	*nbOfBytes*	The length of the data to be converted

Returns

The value converted from the array

10.6.26 void NWKU_TransformValueToArray (uint64_t *value,* uint8_t * *pArray,* uint32_t *nbOfBytes*)

Converts a numeric value to array.

Parameters

in	*value*	The value to be converted
out	*pArray*	The start address of the array
in	*nbOfBytes*	The length of the data to be converted

10.6.27 uint16_t NWKU_Revert16 (uint16_t *value*)

Reverts a 16 bit numeric value.

Parameters

in	*value*	The value to be converted

Returns

The converted value

10.6.28 uint32_t NWKU_Revert32 (uint32_t *value*)

Reverts a 32 bit numeric value.

Parameters

in	*value*	The value to be converted

Returns

The converted value

10.6.29 uint64_t NWKU_Revert64 (uint64_t *value*)

Reverts a 64 bit numeric value.

Parameters

in	*value*	The value to be converted

Returns

The converted value

10.6.30 uint16_t NWKU_TransformArrayToUint16 (uint8_t * *pArray*)

Converts an big endian array to a 16 bit numeric value.

Parameters

in	*pArray*	The start address of the array

Returns

The converted value

10.6.31 uint32_t NWKU_TransformArrayToUint32 (uint8_t * *pArray*)

Converts an big endian array to a 32 bit numeric value.

Parameters

in	*pArray*	The start address of the array

Returns

The converted value

10.6.32 uint64_t NWKU_TransformArrayToUint64 (uint8_t * *pArray*)

Converts an big endian array to a 64 bit numeric value.

Parameters

in	*pArray*	The start address of the array

Returns

The converted value

10.6.33 void NWKU_TransformUint16ToArray (uint8_t * *pArray,* uint16_t *value*)

Converts a 16 bit numeric value to array.

Parameters

in	*value*	The value to be converted
out	*pArray*	The start address of the array

10.6.34 void NWKU_TransformUint32ToArray (uint8_t * *pArray,* uint32_t *value*)

Converts a 32 bit numeric value to array.

Parameters

in	*value*	The value to be converted
out	*pArray*	The start address of the array

10.6.35 void NWKU_TransformUint64ToArray (uint8_t * *pArray,* uint64_t *value*)

Converts a 64 bit numeric value to array.

Parameters

in	*value*	The value to be converted
out	*pArray*	The start address of the array

10.6.36 bool_t NWKU_GetLut8 (lut8_t * *pLutTable,* uint8_t *lutTableSize,* uint8_t *type,* uint8_t * *pEntryIndex*)

Searches an entry in the lookup table indicated by pLutTable.

Parameters

in	*pLutTable*	Pointer to the lookup table
in	*lutTableSize*	Lookup table size
in	*type*	Type to find
out	*pEntryIndex*	Index of the entry in case the entry is found

Returns

TRUE Returned when the entry is found
FALSE Otherwise

10.6.37 int32_t NWKU_atoi (char * *str*)

Converts a string into an integer.

Parameters

in	*pStr*	Pointer to string

Returns

Integer converted from string.

10.6.38 int64_t NWKU_atol (char * *str*)

Converts a string into an long integer.

Parameters

in	*pStr*	pointer to string

Return values

int64_t	integer converted from string.

10.6.39 void NWKU_PrintDec (uint64_t *value*, uint8_t * *pString*, uint32_t *nbOfDigits*, bool_t *bLeadingZeros*)

Prints in a string decimal values.

Parameters

in	*value*	Integer value
	[in/out]	pString Pointer to output location
in	*nbPrintDigits*	Number of digits to be printed
in	*bLeadingZeros*	Indicate if leading zeros are put or omitted TRUE - print leading zeros FALSE - do not print leading zeros

10.6.40 int32_t pton (uint8_t *af*, char * *pTxt*, ipAddr_t * *plpAddr*)

Converts a string into an ipAddr_t. Presentation to network function.

Parameters

in	*af*	Address family(AF_INET, AF_INET6)

in	*pTxt*	Pointer to the start of the string to be parsed
in	*pIpAddr*	Pointer to the start of the allocated ipAddr_t structure

Returns

1 on success
0 string address is not valid
-1 on error

10.6.41 char * ntop (uint8_t *af*, ipAddr_t * *pIpAddr*, char * *pStr*, uint32_t *strLen*)

Converts an ipAddr_t into a string. Network to presentation function.

Parameters

in	*af*	Address family(AF_INET, AF_INET6)
in	*pIpAddr*	Pointer to the start of the allocated ipAddr_t structure
out	*pStr*	Pointer to the allocated string where to put the result
in	*strLen*	Size of the input buffer

Returns

Pointer to the resulted buffer

10.6.42 ptoll (uint8_t * *pIn*, uint32_t *len*, llAddr_t * *pLlAddr*)

Converts a string into an llAddr_t. Presentation to ll function.

Parameters

in	*pIn*	Pointer to the input buffer
in	*len*	Size of the input buffer
in	*pLlAddr*	Pointer to the start of the allocated llAddr_t structure

Returns

TRUE On success
FALSE On error

10.6.43 uint32_t NWKU_AsciiToHex (uint8_t * *pString*, uint32_t *strLen*)

Converts a string into hex.

Parameters

in	*pString*	Pointer to string
in	*strLen*	String length

Returns

Value in hex

10.6.44 uint32_t NWKU_AsciiToDec (uint8_t * *pString*, uint32_t *strLen*)

Converts a string into hex.

Parameters

in	*pString*	Pointer to string
in	*strLen*	String length

Returns

Value in decimal

10.6.45 uint8_t NWKU_ByteToDec (uint8_t *byte*)

Converts a byte from ASCII to decimal.

Parameters

in	*byte*	Byte value in ASCII

Returns

Value in decimal

10.6.46 uint8_t NWKU_NibToAscii (int8_t *nib*, bool_t *useUpperCase*)

Converts a nib from hex to ASCII.

Parameters

in	*nib*	Nib value in hex
in	*useUpperCase*	Flag to specify if conversion is to ASCII uppercase

Returns

Value in ASCII

10.6.47 void NWKU_HexToAscii (uint8_t * *pInputBuff,* uint32_t *inputBuffLen,* uint8_t * *pOutputBuffer,* uint32_t *outputBuffLen,* bool_t *useUpperCase*)

Converts a byte to ASCII.

Parameters

in	*pInputBuff*	Pointer to input buffer
in	*inputBuffLen*	Length of the input buffer
in	*pOutputBuffer*	Pointer to output buffer
in	*outputBuffLen*	Length of the output buffer
in	*useUpperCase*	Indicate if the output shall be in upper/lower case

10.6.48 uint32_t NWKU_TmrRtcGetElapsedTimeInSeconds (uint32_t *timestamp*)

Calculates the time passed in seconds from the provided timestamp.

Parameters

in	*timestamp*	Timestamp in seconds

Returns

Number of seconds that have passed since the provided timestamp

10.6.49 bool_t NWKU_IsNUmber (char * *pString*)

Check if a string is a number.

Parameters

in	*pString*	Pointer to the string

Returns

TRUE If the string represents a number
FALSE If the string does not represent a number

10.6.50 uint32_t NWKU_GetRandomNoFromInterval (uint32_t *startInterval,* uint32_t *endInterval*)

This function returns a random number from a given interval.

Parameters

in	*startInterval*	Start value of the interval
in	*endInterval*	End value of the interval

Returns

Random value

10.6.51 void NWKU_IncrementIp6Addr (ipAddr_t * *pIpAddr*)

This function increments a IPv6 type address

Parameters

in	*pIpAddr*	Pointer to IPv6 address

10.6.52 uint32_t NWKU_RightRotate (uint32_t *val,* uint8_t *amount*)

This function rotates a 32bit number to the right with an amount of bits.

Parameters

in	*val*	Number
in	*amount*	Number of bits to rotate

Returns

Result of the rotation

10.6.53 void NWKU_GetIIDFromLLADDR (llAddr_t * *pLlAddr,* uint16_t *panId,* uint8_t * *pIID*)

The function returns the IID from a Link-Layer address.

Parameters

in	*pLlAddr*	Pointer to the Link-Layer address
in	*panId*	PAN ID
out	*pIID*	Pointer to the variable which will hold the IID

10.6.54 void NWKU_GetLLAddrFromIID (uint8_t * *pIID,* llAddr_t * *pLlAddr*)

This function returns the Link-Layer address from the IID.

Parameters

in	*pIID*	Pointer to the IID
out	*pLlAddr*	Pointer to the variable which will hold the Link-Layer address

10.6.55 void bool_t NWKU_IsIPAddrBasedOnShort (ipAddr_t * *pIpAddr*)

This function returns true if the IPv6 address is formed with short MAC address.

Parameters

in	*pIpAddr*	Pointer to the IPv6 address

Returns

TRUE If address is based on short MAC address
FALSE Otherwise.

10.6.56 bool_t NWKU_GetBit (uint32_t *bitNr,* uint8_t * *pArray*)

This function returns the value of a bit in an array.

Parameters

in	*bitNr*	Bit number in the whole array
in	*pArray*	Pointer to the start of the array

Returns

TRUE If the bit is set
FALSE If the bit is not set

10.6.57 void NWKU_SetBit (uint32_t *bitNr,* uint8_t * *pArray*)

This function sets a bit in an array.

Parameters

in	*bitNr*	Bit number in the whole array
in	*pArray*	Pointer to the start of the array

10.6.58 void NWKU_ClearBit (uint32_t *bitNr,* uint8_t * *pArray*)

This function clears a bit in an array.

Parameters

in	*bitNr*	Bit number in the whole array
in	*pArray*	Pointer to the start of the array

10.6.59 uint32_t NWKU_GetFirstBitValueInRange (uint8_t * *pArray,* uint32_t *lowBitNr,* uint32_t *highBitNr,* bool_t *bitValue*)

This function returns the first bit with value=bitValue in a range in the array.

Parameters

in	*pArray*	Pointer to the start of the array
in	*lowBitNr*	Starting bit number
in	*highBitNr*	Ending bit number
in	*bitValue*	Bit value

Returns

uint32_t Bit number

10.6.60 uint32_t NWKU_GetFirstBitValue (uint8_t * *pArray,* uint32_t *arrayBytes,* bool_t *bitValue*)

This function returns the index of the first bit with value=bitValue.

Parameters

in	*pArray*	Pointer to the start of the array
in	*arrayBytes*	Number of bytes in the array

in	*bitValue*	Bit value

Returns

Bit value

10.6.61 uint32_t NWKU_GetNumOfBits (uint8_t * *pArray,* uint32_t *arrayBytes,* bool_t *bitValue*)

This function returns number of bits of value bitValue from an array

Parameters

in	*pArray*	Pointer to the start of the array
in	*arrayBytes*	Number of bytes in the array
in	*bitValue*	Bit value

Returns

Bit value

10.6.62 uint32_t NWKU_ReverseBits (uint32_t *num*)

Reverse bits

Parameters

in	*num*	The bits to reverse

Returns

The reversed bits

10.6.63 uint32_t NWKU_AddTblEntry (uint32_t *entry,* uint32_t * *pTable,* uint32_t *tableSize*)

This function adds a new entry in a table. The table needs to have uint32_t elements.

Parameters

in	*entry*	Entry value
in	*pTable*	Pointer to the start of the table
in	*tableSize*	The size of the table

Returns

Entry index or -1(0xFFFFFFFF) in case of error

10.6.64 uint32_t NWKU_GetTblEntry (uint32_t *entry,* uint32_t * *pTable,* uint32_t *tableSize*)

This function search for an element in a table.

Parameters

in	*entry*	Entry value
in	*pTable*	Pointer to the start of the table
in	*tableSize*	The size of the table

Returns

Entry index or NULL in case of error

10.6.65 void NWKU_SwapArrayBytes (uint8_t * *pByte,* uint8_t *numOfBytes*)

This function swaps the bytes in an array and puts the result in the same array.

Parameters

	[in/out]	pByte Pointer to the start of the array
in	*numOfBytes*	Size of the array

10.6.66 void NWKU_GenRand (uint8_t * *pRand,* uint8_t *randLen*)

This function generates a random value in the desired array.

Parameters

out	*pRand*	Pointer to the start of the output array
in	*randLen*	Size of the array

10.6.67 uint32_t NWKU_GetTlvLen (uint8_t *type*, uint8_t * *pStart*, uint32_t *len*)

This function returns the length of the TLV type specified.

Parameters

in	*type*	Type identifier for the TLV
in	*pStart*	Pointer to the start if the TLVs
in	*len*	Size of the TLVs buffer

Returns

Length of the specified TLV type

10.6.68 uint8_t * NWKU_GetTlvValue (uint8_t *type*, uint8_t * *pStart*, uint32_t *len*, uint8_t * *pOut*)

This function returns the value of a requested TLV in a list of TLVs. The pointer to the value(if found) will be returned and copied in pOut buffer(if pOut is not NULL).

Parameters

in	*type*	Type identifier for the TLV
in	*pStart*	Pointer to the start if the TLVs
in	*len*	Size of the TLVs buffer
in	*pOut*	Pointer to the output preallocated buffer or NULL

Returns

Pointer to the value of the requested TLV

10.6.69 uint8_t * NWKU_GetTlv (uint8_t *type*, uint8_t * *pStart*, uint32_t *len*, uint8_t * *pOut*, uint32_t * *pOutLen*)

This function returns the start address of the TLV in the pStart buffer.

Parameters

in	*type*	Type identifier for the TLV
in	*pStart*	Pointer to the start if the TLVs
in	*len*	Size of the TLVs buffer
out	*pOut*	If this buffer is provided, the found TLVs will be copied inside
out	*pOutLen*	if this variable is provided, the found TLVs length will be copied

Returns

Pointer to the TLV
NULL if the requested TLV was not found

10.6.70 bool_t NWKU_Pbkdf2 (pbkdf2Params_t * *pInput,* uint8_t * *pOut,* uint32_t *outLen*)

This function calculates pbkdf2 for an input.

Parameters

in	*pInput*	Structure containing the input parameters NOTE: - pInput->pSalt should include the "salt" plus 4 bytes more at the end • pInput->pSalt should specify the "salt" length without the above 4 bytes
in	*pOut*	Pointer to the output
in	*outLen*	Size of the output buffer

Returns

TRUE If the operation has succeeded
FALSE If the operation hasn't succeeded

10.6.71 uint64_t NWKU_GetTimestampMs (void)

Get the timestamp in milliseconds.

Returns

Timestamp in milliseconds

10.6.72 int8_t NWKU_isArrayGreater (const uint8_t * *a*, const uint8_t * *b*, uint8_t *length*)

Compare two numbers represented as array.

Parameters

in	*a*	First array
in	*b*	Second array
in	*length*	How many bytes to compare

Returns

0 - are equal
1 - a > b
-1 - b < a

10.7 Variable Documentation

10.7.1 uint16_t uuint16_t::u16

16bit variable

10.7.2 uint8_t uuint16_t::u8[2]

8bit array

10.7.3 uint32_t uuint32_t::u32

32bit variable

10.7.4 uint16_t uuint32_t::u16[2]

16bit array

10.7.5 uint8_t uuint32_t::u8[4]

8bit array

10.7.6 uint64_t uuint64_t::u64

64bit variable

10.7.7 uint32_t uuint64_t::u32[2]

32bit array

10.7.8 uint16_t uuint64_t::u16[4]

16bit array

10.7.9 uint8_t uuint64_t::u8[8]

8bit array

10.7.10 uint8_t ipAddr_t::addr8[16]

8bit array

10.7.11 uint16_t ipAddr_t::addr16[8]

16bit array

10.7.12 uint32_t ipAddr_t::addr32[4]

32bit array

10.7.13 uint64_t ipAddr_t::addr64[2]

64bit array

10.7.14 struct nwkBuffer_tag* nwkBuffer_t::next

Pointer to next buffer.

10.7.15 uint8_t* nwkBuffer_t::pData

Pointer to data.

10.7.16 uint32_t nwkBuffer_t::size

Size of data.

10.7.17 uint8_t nwkBuffer_t::freeBuffer

Flag used to notify buffer clearance.

10.7.18 uint8_t llAddr_t::eui[8]

Destination address: short/extended.

10.7.19 llAddrSize_t llAddr_t::addrSize

Destination address type: short/extended.

10.7.20 uint8_t ip6Header_t::versionTraficClass

Version Traffic Class.

10.7.21 uint8_t ip6Header_t::trafficClassFlowLabel

Traffic Class Flow label.

10.7.22 uint8_t ip6Header_t::flowLabel[2]

Flow label.

10.7.23 uint8_t ip6Header_t::payloadLength[2]

Payload length.

10.7.24 uint8_t ip6Header_t::nextHeader

Next header.

10.7.25 uint8_t ip6Header_t::hopLimit

Hop limit.

10.7.26 uint8_t ip6Header_t::srcAddr[16]

Source Address.

10.7.27 uint8_t ip6Header_t::dstAddr[16]

Destination Address.

10.7.28 void* ipPktOptions_t::ifHandle

Pointer to interface handler.

10.7.29 nwkBuffer_t* ipPktOptions_t::ipExtensionHeaderBuffer

Pointer to extended options buffer.

10.7.30 void* ipPktOptions_t::ipReassemblyOptions

Pointer to IP reassembly structure.

10.7.31 llAddr_t ipPktOptions_t::srcLlInfo

Source Link Layer information.

10.7.32 uint8_t ipPktOptions_t::ipHdrOffset

Offset from beginning of RX data where IP HDR is found.

10.7.33 uint8_t ipPktOptions_t::hopLimit

Hop limit.

10.7.34 uint8_t ipPktOptions_t::security

Security option.

10.7.35 uint8_t ipPktOptions_t::lqi

Packet LQI.

10.7.36 uint8_t ipPktOptions_t::qos

Packet Quality of Service.

10.7.37 uint8_t ipPktOptions_t::isRelay

Flag to specify if packet is relay.

10.7.38 uint8_t ipPktOptions_t::macSecKeyIdMode

MacSec Key ID Mode.

10.7.39 uint8_t ipPktOptions_t::channel

Packet Channel.

10.7.40 uint16_t ipPktOptions_t::destPanId

Destination PAN ID.

10.7.41 uint16_t ipPktOptions_t::srcPanId

Source PAN ID.

10.7.42 ipIfUniqueId_t recvOptions_t::iplfId

ID of the interface.

10.7.43 uint8_t recvOptions_t::hopLimit

Hop limit.

10.7.44 uint8_t recvOptions_t::security

Security option.

10.7.45 uint8_t recvOptions_t::lqi

Packet LQI.

10.7.46 uint8_t recvOptions_t::isRelay

Flag to specify if packet is relay.

10.7.47 uint8_t recvOptions_t::channel

Packet Channel.

10.7.48 uint8_t recvOptions_t::macSecKeyIdMode

MacSec Key ID Mode.

10.7.49 uint16_t recvOptions_t::macSrcPanId

MAC Source PAN ID.

10.7.50 nwkBuffer_t* ipPktInfo_t::pNwkBuff

Pointer to network buffer.

10.7.51 ipAddr_t* ipPktInfo_t::pIpSrcAddr

Pointer to source IP address.

10.7.52 ipAddr_t* ipPktInfo_t::pIpDstAddr

Pointer to destination IP address.

10.7.53 uint8_t* ipPktInfo_t::pNextProt

Pointer to the next protocol in pNwkBuff->pData.
Do not free this one!

10.7.54 ipAddr_t ipPktInfo_t::ipSrcAddr

Source IP address.

10.7.55 ipAddr_t ipPktInfo_t::ipDstAddr

Destination IP address.

10.7.56 uint32_t { ... } ::nextProtLen

Size of the data of next protocol in pNwkBuff->pData.

10.7.57 uint32_t { ... } ::protocolType

Protocol type.

10.7.58 union { ... } ipPktInfo_t::prot

Protocol information.

10.7.59 uint16_t ipPktInfo_t::srcPort

Source port.

10.7.60 uint16_t ipPktInfo_t::dstPort

Destination port.

10.7.61 ipPktOptions_t ipPktInfo_t::ipPktOptions

IP packet options.

10.7.62 nwkMsgHandler nwkMsg_t::pFunc

Pointer to packet handler.

10.7.63 void* nwkMsg_t::pPload

Pointer to handler payload.

10.7.64 msgQueue_t taskMsgQueue_t::msgQueue

Pointer to task message queue.

10.7.65 osaTaskId_t taskMsgQueue_t::taskId

Pointer to task ID.

10.7.66 osaEventId_t taskMsgQueue_t::taskEventId

Pointer to task event ID.

10.7.67 uint8_t lut8_t::type

Type.

10.7.68 uint8_t lut8_t::idx

Index.

10.7.69 uint8_t nwkStats_t::ipktUsed

IP packets used.

10.7.70 uint8_t nwkStats_t::ipktMax

Maximum IP packets.

10.7.71 uint8_t nwkStats_t::nwkBuffUsed

Network buffers used.

10.7.72 uint8_t nwkStats_t::nwkBuffMax

Maximum network buffers.

10.7.73 uint8_t ipPrefix_t::prefixLen

Size of the prefix in bits.

10.7.74 uint8_t ipPrefix_t::aPrefix[]

Pointer to the start of the prefix.

10.7.75 uint8_t* pbkdf2Params_t::pPass

Pointer to the password.

10.7.76 uint32_t pbkdf2Params_t::passLen

Length of the password.

10.7.77 uint8_t* pbkdf2Params_t::pSalt

Pointer to the salt.

10.7.78 uint32_t pbkdf2Params_t::saltLen

Length of the salt.

10.7.79 uint32_t pbkdf2Params_t::rounds

Number of rounds.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.