OpenThread

- Developing with OpenThread
- Getting Started
  - Overview
  - OpenThread Quick-Start Guide (PDF)
- Fundamentals
  - Overview
  - Thread Fundamentals (PDF)
  - Wireless Networking Fundamentals (PDF)
- OpenThread Developer's Guide
  - Overview
  - Developing and Debugging
    - Overview
    - Configuring Sleepy Devices
      - Sleepy End Devices (SED)
      - Synchronized Sleepy End Devices (SSED)
      - SSED Use Cases
      - Building And Using Silicon Labs Sleepy End Device Demo Applications
    - Single-Band Proprietary Sub-GHz Support with OpenThread (PDF)
    - Using OpenThread with Free RTOS (PDF)
    - Configuring OpenThread Applications for Thread 1.3 (PDF)
  - OpenThread Border Router
    - Overview
    - Using the Silicon Labs RCP with the OpenThread Border Router (PDF)
  - Coexistence
    - Overview
    - Wi-Fi Coexistence Fundamentals (PDF)
    - Zigbee and Thread Coexistence with Wi-Fi (PDF)
    - Configuring Antenna Diversity (PDF)
  - Multiprotocol
    - Overview
    - Multiprotocol Fundamentals (PDF)
    - Dynamic Multiprotocol User's Guide (PDF)
    - Dynamic Multiprotocol Development with Bluetooth and OpenThread on SoCs (PDF)
    - Running Zigbee, OpenThread, and Bluetooth Concurrently on a Linux Host with a Multiprotocol Co-Processor (PDF)
    - Running Zigbee, OpenThread, and Bluetooth Concurrently on a System-on-Chip (PDF)
    - Using the Co-Processor Communication Daemon (CPCd) (PDF)
  - Bootloading

- Overview
- Bootloader Fundamentals (PDF)
- Gecko Bootloader User's Guide (PDF)
- Series 2 Secure Boot with RTSL (PDF)
- Transitioning to the Updated Gecko Bootloader in GSDK 4.0 and Higher (PDF)
- Non-Volatile Memory Use
  - Overview
  - Non-Volatile Data Storage Fundamentals (PDF)
  - Using NVM3 Data Storage (PDF)
- Security
  - Overview
  - IoT Endpoint Security Fundamentals (PDF)
  - Using Silicon Labs Secure Vault Features with OpenThread (PDF)
  - Series 2 Secure Debug (PDF)
  - Production Programming of Series 2 Devices (PDF)
  - Anti-Tamper Protection Configuration and Use (PDF)
  - Authenticating Silicon Labs Devices using Device Certificates (PDF)
  - Secure Key Storage (PDF)
  - Programming Series 2 Devices Using the DCI and SWD (PDF)
  - Series 2 TrustZone (PDF)
  - Integrating Crypto Functionality with PSA Crypto vs. Mbed TLS (PDF)
  - Series 2 TrustZone (PDF)
- Performance
  - Overview
  - Manufacturing Test Overview (PDF)
  - Manufacturing Test Guidelines (PDF)
  - Performance Results for Multi-PAN RCP for OpenThread and Zigbee (PDF)
  - Mesh Network Performance Comparison (PDF)
  - Thread Mesh Network Performance (PDF)
- API Reference Guide
- API Reference
  - Error
    - otError
    - otThreadErrorToString
  - Execution
    - Instance
      - otInstance
      - otChangedFlags
      - otStateChangedCallback
      - otInstanceInit
      - otInstanceInitSingle
      - otInstanceGetId
      - otInstanceIsInitialized
      - otInstanceFinalize

otInstanceGetUptime
otInstanceGetUptimeAsString
otSetStateChangedCallback
otRemoveStateChangeCallback
otInstanceReset
otInstanceResetToBootloader
otInstanceFactoryReset
otInstanceResetRadioStack
otInstanceErasePersistentInfo
otGetVersionString
otGetRadioVersionString
OT_UPTIME_STRING_SIZE
OT_CHANGED_IP6_ADDRESS_ADDED
OT_CHANGED_IP6_ADDRESS_REMOVED
OT_CHANGED_THREAD_ROLE
OT_CHANGED_THREAD_LL_ADDR
OT_CHANGED_THREAD_ML_ADDR
OT_CHANGED_THREAD_RLOC_ADDED
OT_CHANGED_THREAD_RLOC_REMOVED
OT_CHANGED_THREAD_PARTITION_ID
OT_CHANGED_THREAD_KEY_SEQUENCE_COUNTER
OT_CHANGED_THREAD_NETDATA
OT_CHANGED_THREAD_CHILD_ADDED
OT_CHANGED_THREAD_CHILD_REMOVED
OT_CHANGED_IP6_MULTICAST_SUBSCRIBED
OT_CHANGED_IP6_MULTICAST_UNSUBSCRIBED
OT_CHANGED_THREAD_CHANNEL
OT_CHANGED_THREAD_PANID
OT_CHANGED_THREAD_NETWORK_NAME
OT_CHANGED_THREAD_EXT_PANID
OT_CHANGED_NETWORK_KEY
OT_CHANGED_PSKC
OT_CHANGED_SECURITY_POLICY
OT_CHANGED_CHANNEL_MANAGER_NEW_CHANNEL
OT_CHANGED_SUPPORTED_CHANNEL_MASK
OT_CHANGED_COMMISSIONER_STATE
OT_CHANGED_THREAD_NETIF_STATE
OT_CHANGED_THREAD_BACKBONE_ROUTER_STATE
OT_CHANGED_THREAD_BACKBONE_ROUTER_LOCAL
OT_CHANGED_JOINER_STATE
OT_CHANGED_ACTIVE_DATASET
OT_CHANGED_PENDING_DATASET
OT_CHANGED_NAT64_TRANSLATOR_STATE
OT_CHANGED_PARENT_LINK_QUALITY

- Tasklets
  - otTaskletsProcess
  - otTaskletsArePending
  - otTaskletsSignalPending
- IPv6 Networking
  - DNS
    - otDnsTxtEntry
      - mKey
      - mValue
      - mValueLength
    - otDnsTxtEntryIterator
      - mPtr
      - mData
      - mChar
    - otDnsQueryConfig
      - mServerSockAddr
      - mResponseTimeout
      - mMaxTxAttempts
      - mRecursionFlag
      - mNat64Mode
      - mServiceMode
      - mTransportProto
    - otDnsServiceInfo
      - mTtl
      - mPort
      - mPriority
      - mWeight
      - mHostNameBuffer
      - mHostNameBufferSize
      - mHostAddress
      - mHostAddressTtl
      - mTxtData
      - mTxtDataSize
      - mTxtDataTruncated
      - mTxtDataTtl
    - otDnsRecursionFlag
    - otDnsNat64Mode
    - otDnsServiceMode
    - otDnsTransportProto
    - otDnsTxtEntry
    - otDnsTxtEntryIterator
    - otDnsQueryConfig
    - otDnsAddressResponse
    - otDnsAddressCallback

otDnsBrowseResponse
otDnsBrowseCallback
otDnsServiceInfo
otDnsServiceResponse
otDnsServiceCallback
otDnsInitTxtEntryIterator
otDnsGetNextTxtEntry
otDnsEncodeTxtData
otDnsSetNameCompressionEnabled
otDnsIsNameCompressionEnabled
otDnsClientGetDefaultConfig
otDnsClientSetDefaultConfig
otDnsClientResolveAddress
otDnsClientResolveIp4Address
otDnsAddressResponseGetHostName
otDnsAddressResponseGetAddress
otDnsClientBrowse
otDnsBrowseResponseGetServiceName
otDnsBrowseResponseGetServiceInstance
otDnsBrowseResponseGetServiceInfo
otDnsBrowseResponseGetHostAddress
otDnsClientResolveService
otDnsClientResolveServiceAndHostAddress
otDnsServiceResponseGetServiceName
otDnsServiceResponseGetServiceInfo
otDnsServiceResponseGetHostAddress
OT_DNS_MAX_NAME_SIZE
OT_DNS_MAX_LABEL_SIZE
OT_DNS_TXT_KEY_MIN_LENGTH
OT_DNS_TXT_KEY_MAX_LENGTH
DNS-SD Server
otDnssdServiceInstanceInfo
mFullName
mHostName
mAddressNum
mAddresses
mPort
mPriority
mWeight
mTxtLength
mTxtData
mTtl
otDnssdHostInfo
mAddressNum

mAddresses
mTtl
otDnssdCounters
mSuccessResponse
mServerFailureResponse
mFormatErrorResponse
mNameErrorResponse
mNotImplementedResponse
mOtherResponse
mResolvedBySrp
otDnssdQueryType
otDnssdQuerySubscribeCallback
otDnssdQueryUnsubscribeCallback
otDnssdQuery
otDnssdServiceInstanceInfo
otDnssdHostInfo
otDnssdCounters
otDnssdQuerySetCallbacks
otDnssdQueryHandleDiscoveredServiceInstance
otDnssdQueryHandleDiscoveredHost
otDnssdGetNextQuery
otDnssdGetQueryTypeAndName
otDnssdGetCounters
otDnssdUpstreamQuerySetEnabled
otDnssdUpstreamQueryIsEnabled
ICMPv6
otIcmp6Header
otIcmp6Header::OT_TOOL_PACKED_FIELD
m8
m16
m32
mType
mCode
mChecksum
mData
otIcmp6Handler
mReceiveCallback
mContext
mNext
otIcmp6Type
otIcmp6Code
otIcmp6EchoMode
otIcmp6Type
otIcmp6Code

otIcmp6Header
otIcmp6ReceiveCallback
otIcmp6Handler
otIcmp6EchoMode
OT_TOOL_PACKED_END
otIcmp6GetEchoMode
otIcmp6SetEchoMode
otIcmp6RegisterHandler
otIcmp6SendEchoRequest
OT_ICMP6_HEADER_DATA_SIZE
OT_ICMP6_ROUTER_ADVERT_MIN_SIZE
IPv6
otIp6InterfaceIdentifier
otIp6InterfaceIdentifier::OT_TOOL_PACKED_FIELD
m8
m16
m32
mFields
otIp6NetworkPrefix
m8
otIp6AddressComponents
mNetworkPrefix
mIid
otIp6Address
otIp6Address::OT_TOOL_PACKED_FIELD
m8
m16
m32
mComponents
mFields
otIp6Prefix
mPrefix
mLength
otNetifAddress
mAddress
mPrefixLength
mAddressOrigin
mPreferred
mValid
mScopeOverrideValid
mScopeOverride
mRloc
mMeshLocal
mNext

otNetifMulticastAddress
mAddress
mNext
otSockAddr
mAddress
mPort
otMessageInfo
mSockAddr
mPeerAddr
mSockPort
mPeerPort
mLinkInfo
mHopLimit
mEcn
mIsHostInterface
mAllowZeroHopLimit
mMulticastLoop
otIp6AddressInfo
mAddress
mPrefixLength
mScope
mPreferred
otPacketsAndBytes
mPackets
mBytes
otBorderRoutingCounters
mInboundUnicast
mInboundMulticast
mOutboundUnicast
mOutboundMulticast
mRaRx
mRaTxSuccess
mRaTxFailure
mRsRx
mRsTxSuccess
mRsTxFailure
@2
@3
@4
otIp6InterfaceIdentifier
otIp6NetworkPrefix
otIp6AddressComponents
otIp6Address
otIp6Prefix

otNetifAddress
otNetifMulticastAddress
otSockAddr
otMessageInfo
otIp6ReceiveCallback
otIp6AddressInfo
otIp6AddressCallback
otIp6SlaacPrefixFilter
otIp6RegisterMulticastListenersCallback
otPacketsAndBytes
otBorderRoutingCounters
OT_TOOL_PACKED_END
otIp6SetEnabled
otIp6IsEnabled
otIp6AddUnicastAddress
otIp6RemoveUnicastAddress
otIp6GetUnicastAddresses
otIp6HasUnicastAddress
otIp6SubscribeMulticastAddress
otIp6UnsubscribeMulticastAddress
otIp6GetMulticastAddresses
otIp6IsMulticastPromiscuousEnabled
otIp6SetMulticastPromiscuousEnabled
otIp6NewMessage
otIp6NewMessageFromBuffer
otIp6SetReceiveCallback
otIp6SetAddressCallback
otIp6IsReceiveFilterEnabled
otIp6SetReceiveFilterEnabled
otIp6Send
otIp6AddUnsecurePort
otIp6RemoveUnsecurePort
otIp6RemoveAllUnsecurePorts
otIp6GetUnsecurePorts
otIp6IsAddressEqual
otIp6ArePrefixesEqual
otIp6AddressFromString
otIp6PrefixFromString
otIp6AddressToString
otIp6SockAddrToString
otIp6PrefixToString
otIp6PrefixMatch
otIp6GetPrefix
otIp6IsAddressUnspecified

otIp6SelectSourceAddress
otIp6IsSlaacEnabled
otIp6SetSlaacEnabled
otIp6SetSlaacPrefixFilter
otIp6RegisterMulticastListeners
otIp6SetMeshLocalId
otIp6ProtoToString
otIp6GetBorderRoutingCounters
otIp6ResetBorderRoutingCounters
OT_IP6_PREFIX_SIZE
OT_IP6_PREFIX_BITSIZE
OT_IP6_IID_SIZE
OT_IP6_ADDRESS_SIZE
OT_IP6_HEADER_SIZE
OT_IP6_HEADER_PROTO_OFFSET
OT_IP6_ADDRESS_STRING_SIZE
OT_IP6_SOCK_ADDR_STRING_SIZE
OT_IP6_PREFIX_STRING_SIZE
OT_IP6_MAX_MLR_ADDRESSES
NAT64
otIp4Address
otIp4Address::OT_TOOL_PACKED_FIELD
m8
m32
mFields
otIp4Cidr
mAddress
mLength
otNat64Counters
m4To6Packets
m4To6Bytes
m6To4Packets
m6To4Bytes
otNat64ProtocolCounters
mTotal
mIcmp
mUdp
mTcp
otNat64ErrorCounters
mCount4To6
mCount6To4
otNat64AddressMapping
mId
mIp4

mIp6
mRemainingTimeMs
mCounters
otNat64AddressMappingIterator
mPtr
otNat64DropReason
otNat64State
otIp4Address
otIp4Cidr
otNat64Counters
otNat64ProtocolCounters
otNat64DropReason
otNat64ErrorCounters
otNat64AddressMapping
otNat64AddressMappingIterator
otNat64ReceiveIp4Callback
OT_TOOL_PACKED_END
otNat64GetCounters
otNat64GetErrorCounters
otNat64InitAddressMappingIterator
otNat64GetNextAddressMapping
otNat64GetTranslatorState
otNat64GetPrefixManagerState
otNat64SetEnabled
otIp4NewMessage
otNat64SetIp4Cidr
otNat64Send
otNat64SetReceiveIp4Callback
otNat64GetCidr
otIp4IsAddressEqual
otIp4ExtractFromIp6Address
otIp4AddressToString
otIp4CidrFromString
otIp4CidrToString
otIp4AddressFromString
otNat64SynthesizeIp6Address
OT_IP4_ADDRESS_SIZE
OT_IP4_ADDRESS_STRING_SIZE
OT_IP4_CIDR_STRING_SIZE
SRP
otSrpClientHostInfo
mName
mAddresses
mNumAddresses

mAutoAddress
mState
otSrpClientService
mName
mInstanceName
mSubTypeLabels
mTxtEntries
mPort
mPriority
mWeight
mNumTxtEntries
mState
mData
mNext
mLease
mKeyLease
otSrpClientBuffersServiceEntry
mService
mTxtEntry
otSrpServerTtlConfig
mMinTtl
mMaxTtl
otSrpServerLeaseConfig
mMinLease
mMaxLease
mMinKeyLease
mMaxKeyLease
otSrpServerLeaseInfo
mLease
mKeyLease
mRemainingLease
mRemainingKeyLease
otSrpServerResponseCounters
mSuccess
mServerFailure
mFormatError
mNameExists
mRefused
mOther
otSrpClientItemState
otSrpServerState
otSrpServerAddressMode
otSrpClientHostInfo
otSrpClientService

otSrpClientCallback
otSrpClientAutoStartCallback
otSrpClientBuffersServiceEntry
otSrpServerHost
otSrpServerService
otSrpServerServiceUpdateId
otSrpServerAddressMode
otSrpServerTtlConfig
otSrpServerLeaseConfig
otSrpServerLeaseInfo
otSrpServerResponseCounters
otSrpServerServiceUpdateHandler
otSrpClientStart
otSrpClientStop
otSrpClientIsRunning
otSrpClientGetServerAddress
otSrpClientSetCallback
otSrpClientEnableAutoStartMode
otSrpClientDisableAutoStartMode
otSrpClientIsAutoStartModeEnabled
otSrpClientGetTtl
otSrpClientSetTtl
otSrpClientGetLeaseInterval
otSrpClientSetLeaseInterval
otSrpClientGetKeyLeaseInterval
otSrpClientSetKeyLeaseInterval
otSrpClientGetHostInfo
otSrpClientSetHostName
otSrpClientEnableAutoHostAddress
otSrpClientSetHostAddresses
otSrpClientAddService
otSrpClientRemoveService
otSrpClientClearService
otSrpClientGetServices
otSrpClientRemoveHostAndServices
otSrpClientClearHostAndServices
otSrpClientGetDomainName
otSrpClientSetDomainName
otSrpClientItemStateToString
otSrpClientSetServiceKeyRecordEnabled
otSrpClientIsServiceKeyRecordEnabled
otSrpClientBuffersGetHostNameString
otSrpClientBuffersGetHostAddressesArray
otSrpClientBuffersAllocateService

otSrpClientBuffersFreeService
otSrpClientBuffersFreeAllServices
otSrpClientBuffersGetServiceEntryServiceNameString
otSrpClientBuffersGetServiceEntryInstanceNameString
otSrpClientBuffersGetServiceEntryTxtBuffer
otSrpClientBuffersGetSubTypeLabelsArray
otSrpServerGetDomain
otSrpServerSetDomain
otSrpServerGetState
otSrpServerGetPort
otSrpServerGetAddressMode
otSrpServerSetAddressMode
otSrpServerGetAnycastModeSequenceNumber
otSrpServerSetAnycastModeSequenceNumber
otSrpServerSetEnabled
otSrpServerSetAutoEnableMode
otSrpServerIsAutoEnableMode
otSrpServerGetTtlConfig
otSrpServerSetTtlConfig
otSrpServerGetLeaseConfig
otSrpServerSetLeaseConfig
otSrpServerSetServiceUpdateHandler
otSrpServerHandleServiceUpdateResult
otSrpServerGetNextHost
otSrpServerGetResponseCounters
otSrpServerHostIsDeleted
otSrpServerHostGetFullName
otSrpServerHostMatchesFullName
otSrpServerHostGetAddresses
otSrpServerHostGetLeaseInfo
otSrpServerHostGetNextService
otSrpServerServiceIsDeleted
otSrpServerServiceGetInstanceName
otSrpServerServiceMatchesInstanceName
otSrpServerServiceGetInstanceLabel
otSrpServerServiceGetServiceName
otSrpServerServiceMatchesServiceName
otSrpServerServiceGetNumberOfSubTypes
otSrpServerServiceGetSubTypeServiceNameAt
otSrpServerServiceHasSubTypeServiceName
otSrpServerParseSubTypeServiceName
otSrpServerServiceGetPort
otSrpServerServiceGetWeight

otSrpServerServiceGetPriority
otSrpServerServiceGetTtl
otSrpServerServiceGetTxtData
otSrpServerServiceGetHost
otSrpServerServiceGetLeaseInfo
Ping Sender
otPingSenderReply
mSenderAddress
mRoundTripTime
mSize
mSequenceNumber
mHopLimit
otPingSenderStatistics
mSentCount
mReceivedCount
mTotalRoundTripTime
mMinRoundTripTime
mMaxRoundTripTime
mIsMulticast
otPingSenderConfig
mSource
mDestination
mReplyCallback
mStatisticsCallback
mCallbackContext
mSize
mCount
mInterval
mTimeout
mHopLimit
mAllowZeroHopLimit
mMulticastLoop
otPingSenderReply
otPingSenderStatistics
otPingSenderReplyCallback
otPingSenderStatisticsCallback
otPingSenderConfig
otPingSenderPing
otPingSenderStop
TCP
TCP
otLinkedBuffer
mNext
mData

mLength
otTcpEndpoint
mSize
mAlign
mTcb
mNext
mContext
mEstablishedCallback
mSendDoneCallback
mForwardProgressCallback
mReceiveAvailableCallback
mDisconnectedCallback
mTimers
mReceiveLinks
mSockAddr
mPendingCallbacks
otTcpEndpointInitializeArgs
mContext
mEstablishedCallback
mSendDoneCallback
mForwardProgressCallback
mReceiveAvailableCallback
mDisconnectedCallback
mReceiveBuffer
mReceiveBufferSize
otTcpListener
mSize
mAlign
mTcbListen
mNext
mContext
mAcceptReadyCallback
mAcceptDoneCallback
otTcpListenerInitializeArgs
mContext
mAcceptReadyCallback
mAcceptDoneCallback
otTcpDisconnectedReason
@22
@23
otTcpIncomingConnectionAction
otLinkedBuffer
otTcpEndpoint
otTcpEstablished

otTcpSendDone
otTcpForwardProgress
otTcpReceiveAvailable
otTcpDisconnectedReason
otTcpDisconnected
otTcpEndpointInitializeArgs
otTcpListener
otTcpIncomingConnectionAction
otTcpAcceptReady
otTcpAcceptDone
otTcpListenerInitializeArgs
otTcpEndpointInitialize
otTcpEndpointGetInstance
otTcpEndpointGetContext
otTcpGetLocalAddress
otTcpGetPeerAddress
otTcpBind
otTcpConnect
otTcpSendByReference
otTcpSendByExtension
otTcpReceiveByReference
otTcpReceiveContiguify
otTcpCommitReceive
otTcpSendEndOfStream
otTcpAbort
otTcpEndpointDeinitialize
otTcpListenerInitialize
otTcpListenerGetInstance
otTcpListenerGetContext
otTcpListen
otTcpStopListening
otTcpListenerDeinitialize
OT_TCP_ENDPOINT_TCB_SIZE_BASE
OT_TCP_ENDPOINT_TCB_NUM_PTR
OT_TCP_RECEIVE_BUFFER_SIZE_FEW_HOPS
OT_TCP_RECEIVE_BUFFER_SIZE_MANY_HOPS
OT_TCP_LISTENER_TCB_SIZE_BASE
OT_TCP_LISTENER_TCB_NUM_PTR
TCP Abstractions
otTcpCircularSendBuffer
mDataBuffer
mCapacity
mStartIndex
mCapacityUsed

mSendLinks
mFirstSendLinkIndex
otTcpEndpointAndCircularSendBuffer
mEndpoint
mSendBuffer
@26
otTcpCircularSendBuffer
otTcpEndpointAndCircularSendBuffer
otTcpCircularSendBufferInitialize
otTcpCircularSendBufferWrite
otTcpCircularSendBufferHandleForwardProgress
otTcpCircularSendBufferGetFreeSpace
otTcpCircularSendBufferForceDiscardAll
otTcpCircularSendBufferDeinitialize
otTcpMbedTlsSslSendCallback
otTcpMbedTlsSslRecvCallback
UDP
UDP
otUdpReceiver
mNext
mHandler
mContext
otUdpSocket
mSockName
mPeerName
mHandler
mContext
mHandle
mNext
otNetifIdentifier
otUdpHandler
otUdpReceiver
otUdpReceive
otUdpSocket
otNetifIdentifier
otUdpAddReceiver
otUdpRemoveReceiver
otUdpSendDatagram
otUdpNewMessage
otUdpOpen
otUdpIsOpen
otUdpClose
otUdpBind
otUdpConnect

- otUdpSend
- otUdpGetSockets
- UDP Forward
  - otUdpForwarder
  - otUdpForwardSetForwarder
  - otUdpForwardReceive
  - otUdpIsPortInUse
- Link
  - Link
    - otThreadLinkInfo
      - mPanId
      - mChannel
      - mRss
      - mLqi
      - mLinkSecurity
      - mIsDstPanIdBroadcast
      - mTimeSyncSeq
      - mNetworkTimeOffset
      - mRadioType
    - otMacFilterEntry
      - mExtAddress
      - mRssIn
    - otMacCounters
      - mTxTotal
      - mTxUnicast
      - mTxBroadcast
      - mTxAckRequested
      - mTxAcked
      - mTxNoAckRequested
      - mTxData
      - mTxDataPoll
      - mTxBeacon
      - mTxBeaconRequest
      - mTxOther
      - mTxRetry
      - mTxDirectMaxRetryExpiry
      - mTxIndirectMaxRetryExpiry
      - mTxErrCca
      - mTxErrAbort
      - mTxErrBusyChannel
      - mRxTotal
      - mRxUnicast
      - mRxBroadcast
      - mRxData

mRxDataPoll
mRxBeacon
mRxBeaconRequest
mRxOther
mRxAddressFiltered
mRxDestAddrFiltered
mRxDuplicated
mRxErrNoFrame
mRxErrUnknownNeighbor
mRxErrInvalidSrcAddr
mRxErrSec
mRxErrFcs
mRxErrOther
otActiveScanResult
mExtAddress
mNetworkName
mExtendedPanId
mSteeringData
mPanId
mJoinerUdpPort
mChannel
mRssi
mLqi
mVersion
mIsNative
mDiscover
mIsJoinable
otEnergyScanResult
mChannel
mMaxRssi
otMacFilterAddressMode
otThreadLinkInfo
otMacFilterIterator
otMacFilterAddressMode
otMacFilterEntry
otMacCounters
otActiveScanResult
otEnergyScanResult
otHandleActiveScanResult
otHandleEnergyScanResult
otLinkPcapCallback
otLinkActiveScan
otLinkIsActiveScanInProgress
otLinkEnergyScan

otLinkIsEnergyScanInProgress
otLinkSendDataRequest
otLinkIsInTransmitState
otLinkGetChannel
otLinkSetChannel
otLinkGetSupportedChannelMask
otLinkSetSupportedChannelMask
otLinkGetExtendedAddress
otLinkSetExtendedAddress
otLinkGetFactoryAssignedIeeeEui64
otLinkGetPanId
otLinkSetPanId
otLinkGetPollPeriod
otLinkSetPollPeriod
otLinkGetShortAddress
otLinkGetMaxFrameRetriesDirect
otLinkSetMaxFrameRetriesDirect
otLinkGetMaxFrameRetriesIndirect
otLinkSetMaxFrameRetriesIndirect
otLinkFilterGetAddressMode
otLinkFilterSetAddressMode
otLinkFilterAddAddress
otLinkFilterRemoveAddress
otLinkFilterClearAddresses
otLinkFilterGetNextAddress
otLinkFilterAddRssIn
otLinkFilterRemoveRssIn
otLinkFilterSetDefaultRssIn
otLinkFilterClearDefaultRssIn
otLinkFilterClearAllRssIn
otLinkFilterGetNextRssIn
otLinkSetRadioFilterEnabled
otLinkIsRadioFilterEnabled
otLinkConvertRssToLinkQuality
otLinkConvertLinkQualityToRss
otLinkGetTxDirectRetrySuccessHistogram
otLinkGetTxIndirectRetrySuccessHistogram
otLinkResetTxRetrySuccessHistogram
otLinkGetCounters
otLinkResetCounters
otLinkSetPcapCallback
otLinkIsPromiscuous
otLinkSetPromiscuous
otLinkGetCslChannel

- otLinkSetCslChannel
- otLinkGetCslPeriod
- otLinkSetCslPeriod
- otLinkGetCslTimeout
- otLinkSetCslTimeout
- otLinkGetCcaFailureRate
- otLinkSetEnabled
- otLinkIsEnabled
- otLinkIsCslEnabled
- otLinkIsCslSupported
- otLinkSendEmptyData
- otLinkSetRegion
- otLinkGetRegion
- OT_US_PER_TEN_SYMBOLS
- OT_MAC_FILTER_FIXED_RSS_DISABLED
- OT_MAC_FILTER_ITERATOR_INIT
- OT_LINK_CSL_PERIOD_TEN_SYMBOLS_UNIT_IN_USEC

Link Metrics

- otLinkMetricsValues
  - mMetrics
  - mPduCountValue
  - mLqiValue
  - mLinkMarginValue
  - mRssiValue
- otLinkMetricsSeriesFlags
  - mLinkProbe
  - mMacData
  - mMacDataRequest
  - mMacAck
- otLinkMetricsEnhAckFlags
- otLinkMetricsStatus
- otLinkMetricsValues
- otLinkMetricsSeriesFlags
- otLinkMetricsEnhAckFlags
- otLinkMetricsStatus
- otLinkMetricsReportCallback
- otLinkMetricsMgmtResponseCallback
- otLinkMetricsEnhAckProbingIeReportCallback
- otLinkMetricsQuery
- otLinkMetricsConfigForwardTrackingSeries
- otLinkMetricsConfigEnhAckProbing
- otLinkMetricsSendLinkProbe
- otLinkMetricsManagerSetEnabled
- otLinkMetricsManagerGetMetricsValueByExtAddr

Raw Link
otLinkRawReceiveDone
otLinkRawTransmitDone
otLinkRawEnergyScanDone
otLinkRawSetReceiveDone
otLinkRawIsEnabled
otLinkRawGetPromiscuous
otLinkRawSetPromiscuous
otLinkRawSetShortAddress
otLinkRawSleep
otLinkRawReceive
otLinkRawIsTransmittingOrScanning
otLinkRawGetTransmitBuffer
otLinkRawTransmit
otLinkRawGetRssi
otLinkRawGetCaps
otLinkRawEnergyScan
otLinkRawSrcMatchEnable
otLinkRawSrcMatchAddShortEntry
otLinkRawSrcMatchAddExtEntry
otLinkRawSrcMatchClearShortEntry
otLinkRawSrcMatchClearExtEntry
otLinkRawSrcMatchClearShortEntries
otLinkRawSrcMatchClearExtEntries
otLinkRawSetMacKey
otLinkRawSetMacFrameCounter
otLinkRawSetMacFrameCounterIfLarger
otLinkRawGetRadioTime
Message
otMessageSettings
mLinkSecurityEnabled
mPriority
otMessageQueue
mData
otMessageQueueInfo
mNumMessages
mNumBuffers
mTotalBytes
otBufferInfo
mTotalBuffers
mFreeBuffers
mMaxUsedBuffers
m6loSendQueue
m6loReassemblyQueue

mIp6Queue
mMplQueue
mMleQueue
mCoapQueue
mCoapSecureQueue
mApplicationCoapQueue
otMessagePriority
otMessageOrigin
otMessage
otMessagePriority
otMessageOrigin
otMessageSettings
otMessageQueueInfo
otBufferInfo
otMessageFree
otMessageGetLength
otMessageSetLength
otMessageGetOffset
otMessageSetOffset
otMessageIsLinkSecurityEnabled
otMessageIsLoopbackToHostAllowed
otMessageSetLoopbackToHostAllowed
otMessageGetOrigin
otMessageSetOrigin
otMessageSetDirectTransmission
otMessageGetRss
otMessageAppend
otMessageRead
otMessageWrite
otMessageQueueInit
otMessageQueueEnqueue
otMessageQueueEnqueueAtHead
otMessageQueueDequeue
otMessageQueueGetHead
otMessageQueueGetNext
otMessageGetBufferInfo
otMessageResetBufferInfo
Multi Radio Link
otRadioLinkInfo
mPreference
otMultiRadioNeighborInfo
mSupportsIeee802154
mSupportsTrelUdp6
mIeee802154Info

- mTrelUdp6Info
- otRadioLinkInfo
- otMultiRadioNeighborInfo
- otMultiRadioGetNeighborInfo

TREL - Thread Stack

- otTrelPeer
  - mExtAddress
  - mExtPanId
  - mSockAddr
- otTrelPeer
- otTrelPeerIterator
- otTrelSetEnabled
- otTrelIsEnabled
- otTrelInitPeerIterator
- otTrelGetNextPeer
- otTrelSetFilterEnabled
- otTrelIsFilterEnabled

Thread

- Backbone Router
  - otBackboneRouterConfig
    - mServer16
    - mReregistrationDelay
    - mMlrTimeout
    - mSequenceNumber
  - otBackboneRouterMulticastListenerInfo
    - mAddress
    - mTimeout
  - otBackboneRouterNdProxyInfo
    - mMeshLocalIid
    - mTimeSinceLastTransaction
    - mRloc16
  - otBackboneRouterState
  - otBackboneRouterMulticastListenerEvent
  - otBackboneRouterNdProxyEvent
  - otBackboneRouterDomainPrefixEvent
  - otBackboneRouterConfig
  - otBackboneRouterMulticastListenerCallback
  - otBackboneRouterMulticastListenerIterator
  - otBackboneRouterMulticastListenerInfo
  - otBackboneRouterNdProxyCallback
  - otBackboneRouterNdProxyInfo
  - otBackboneRouterDomainPrefixCallback
  - otBackboneRouterGetPrimary
  - otBackboneRouterSetEnabled

otBackboneRouterGetState
otBackboneRouterGetConfig
otBackboneRouterSetConfig
otBackboneRouterRegister
otBackboneRouterGetRegistrationJitter
otBackboneRouterSetRegistrationJitter
otBackboneRouterGetDomainPrefix
otBackboneRouterConfigNextDuaRegistrationResponse
otBackboneRouterConfigNextMulticastListenerRegistrationResponse
otBackboneRouterSetMulticastListenerCallback
otBackboneRouterMulticastListenerClear
otBackboneRouterMulticastListenerAdd
otBackboneRouterMulticastListenerGetNext
otBackboneRouterSetNdProxyCallback
otBackboneRouterGetNdProxyInfo
otBackboneRouterSetDomainPrefixCallback
OT_BACKBONE_ROUTER_MULTICAST_LISTENER_ITERATOR_INIT
Border Agent
otBorderAgentId
mId
otBorderAgentState
otBorderAgentId
otBorderAgentState
OT_TOOL_PACKED_END
otBorderAgentGetState
otBorderAgentGetUdpPort
otBorderAgentGetId
otBorderAgentSetId
OT_BORDER_AGENT_ID_LENGTH
Border Router
otBorderRouterNetDataFullCallback
otBorderRouterGetNetData
otBorderRouterAddOnMeshPrefix
otBorderRouterRemoveOnMeshPrefix
otBorderRouterGetNextOnMeshPrefix
otBorderRouterAddRoute
otBorderRouterRemoveRoute
otBorderRouterGetNextRoute
otBorderRouterRegister
otBorderRouterSetNetDataFullCallback
Border Routing Manager
otBorderRoutingPrefixTableIterator
mPtr1
mPtr2

mData32
otBorderRoutingPrefixTableEntry
mRouterAddress
mPrefix
mIsOnLink
mMsecSinceLastUpdate
mValidLifetime
mRoutePreference
mPreferredLifetime
otBorderRoutingState
otBorderRoutingDhcp6PdState
otBorderRoutingPrefixTableIterator
otBorderRoutingPrefixTableEntry
otBorderRoutingInit
otBorderRoutingSetEnabled
otBorderRoutingGetState
otBorderRoutingGetRouteInfoOptionPreference
otBorderRoutingSetRouteInfoOptionPreference
otBorderRoutingClearRouteInfoOptionPreference
otBorderRoutingGetRoutePreference
otBorderRoutingSetRoutePreference
otBorderRoutingClearRoutePreference
otBorderRoutingGetOmrPrefix
otBorderRoutingGetPdOmrPrefix
otBorderRoutingGetFavoredOmrPrefix
otBorderRoutingGetOnLinkPrefix
otBorderRoutingGetFavoredOnLinkPrefix
otBorderRoutingGetNat64Prefix
otBorderRoutingGetFavoredNat64Prefix
otBorderRoutingPrefixTableInitIterator
otBorderRoutingGetNextPrefixTableEntry
otBorderRoutingDhcp6PdSetEnabled
Commissioner
otSteeringData
mLength
m8
otCommissioningDataset
mLocator
mSessionId
mSteeringData
mJoinerUdpPort
mIsLocatorSet
mIsSessionIdSet
mIsSteeringDataSet

mIsJoinerUdpPortSet
mHasExtraTlv
otJoinerPskd
m8
otJoinerInfo
mType
mEui64
mDiscerner
mSharedId
mPskd
mExpirationTime
otCommissionerState
otCommissionerJoinerEvent
otJoinerInfoType
otCommissionerState
otCommissionerJoinerEvent
otSteeringData
otCommissioningDataset
otJoinerPskd
otJoinerInfoType
otJoinerInfo
otCommissionerStateCallback
otCommissionerJoinerCallback
otCommissionerEnergyReportCallback
otCommissionerPanIdConflictCallback
otCommissionerStart
otCommissionerStop
otCommissionerGetId
otCommissionerSetId
otCommissionerAddJoiner
otCommissionerAddJoinerWithDiscerner
otCommissionerGetNextJoinerInfo
otCommissionerRemoveJoiner
otCommissionerRemoveJoinerWithDiscerner
otCommissionerGetProvisioningUrl
otCommissionerSetProvisioningUrl
otCommissionerAnnounceBegin
otCommissionerEnergyScan
otCommissionerPanIdQuery
otCommissionerSendMgmtGet
otCommissionerSendMgmtSet
otCommissionerGetSessionId
otCommissionerGetState
OT_COMMISSIONING_PASSPHRASE_MIN_SIZE

OT_COMMISSIONING_PASSPHRASE_MAX_SIZE
OT_PROVISIONING_URL_MAX_SIZE
OT_STEERING_DATA_MAX_LENGTH
OT_JOINER_MAX_PSKD_LENGTH
General
otBorderRouterConfig
mPrefix
mPreference
mPreferred
mSlaac
mDhcp
mConfigure
mDefaultRoute
mOnMesh
mStable
mNdDns
mDp
mRloc16
otLowpanContextInfo
mContextId
mCompressFlag
mPrefix
otExternalRouteConfig
mPrefix
mRloc16
mPreference
mNat64
mStable
mNextHopIsThisDevice
mAdvPio
otServerConfig
mStable
mServerDataLength
mServerData
mRloc16
otServiceConfig
mServiceId
mEnterpriseNumber
mServiceDataLength
mServiceData
mServerConfig
otNetworkDiagConnectivity
mParentPriority
mLinkQuality3

mLinkQuality2
mLinkQuality1
mLeaderCost
mIdSequence
mActiveRouters
mSedBufferSize
mSedDatagramCount
otNetworkDiagRouteData
mRouterId
mLinkQualityOut
mLinkQualityIn
mRouteCost
otNetworkDiagRoute
mIdSequence
mRouteCount
mRouteData
otNetworkDiagMacCounters
mIfInUnknownProtos
mIfInErrors
mIfOutErrors
mIfInUcastPkts
mIfInBroadcastPkts
mIfInDiscards
mIfOutUcastPkts
mIfOutBroadcastPkts
mIfOutDiscards
otNetworkDiagMleCounters
mDisabledRole
mDetachedRole
mChildRole
mRouterRole
mLeaderRole
mAttachAttempts
mPartitionIdChanges
mBetterPartitionAttachAttempts
mParentChanges
mTrackedTime
mDisabledTime
mDetachedTime
mChildTime
mRouterTime
mLeaderTime
otNetworkDiagChildEntry
mTimeout

mLinkQuality
mChildId
mMode
otNetworkDiagTlv
mType
mExtAddress
mAddr16
mMode
mTimeout
mConnectivity
mRoute
mLeaderData
mMacCounters
mMleCounters
mBatteryLevel
mSupplyVoltage
mMaxChildTimeout
mVersion
mVendorName
mVendorModel
mVendorSwVersion
mThreadStackVersion
mCount
m8
mNetworkData
mList
mIp6AddrList
mTable
mChildTable
mChannelPages
mData
otLinkModeConfig
mRxOnWhenIdle
mDeviceType
mNetworkData
otNeighborInfo
mExtAddress
mAge
mConnectionTime
mRloc16
mLinkFrameCounter
mMleFrameCounter
mLinkQualityIn
mAverageRssi

mLastRssi
mLinkMargin
mFrameErrorRate
mMessageErrorRate
mVersion
mRxOnWhenIdle
mFullThreadDevice
mFullNetworkData
mIsChild
otLeaderData
mPartitionId
mWeighting
mDataVersion
mStableDataVersion
mLeaderRouterId
otRouterInfo
mExtAddress
mRloc16
mRouterId
mNextHop
mPathCost
mLinkQualityIn
mLinkQualityOut
mAge
mAllocated
mLinkEstablished
mVersion
mCslClockAccuracy
mCslUncertainty
otIpCounters
mTxSuccess
mRxSuccess
mTxFailure
mRxFailure
otMleCounters
mDisabledRole
mDetachedRole
mChildRole
mRouterRole
mLeaderRole
mAttachAttempts
mPartitionIdChanges
mBetterPartitionAttachAttempts
mDisabledTime

mDetachedTime
mChildTime
mRouterTime
mLeaderTime
mTrackedTime
mParentChanges
otThreadParentResponseInfo
mExtAddr
mRloc16
mRssi
mPriority
mLinkQuality3
mLinkQuality2
mLinkQuality1
mIsAttached
otThreadDiscoveryRequestInfo
mExtAddress
mVersion
mIsJoiner
otRoutePreference
otNetDataPublisherEvent
@5
otDeviceRole
otNetworkDataIterator
otBorderRouterConfig
otLowpanContextInfo
otExternalRouteConfig
otRoutePreference
otServerConfig
otServiceConfig
otNetDataPublisherEvent
otNetDataDnsSrpServicePublisherCallback
otNetDataPrefixPublisherCallback
otNetworkDiagIterator
otNetworkDiagConnectivity
otNetworkDiagRouteData
otNetworkDiagRoute
otNetworkDiagMacCounters
otNetworkDiagMleCounters
otNetworkDiagChildEntry
otNetworkDiagTlv
otReceiveDiagnosticGetCallback
otLinkModeConfig
otNeighborInfoIterator

otLeaderData
otIpCounters
otMleCounters
otThreadParentResponseInfo
otDetachGracefullyCallback
otThreadParentResponseCallback
otThreadDiscoveryRequestInfo
otThreadDiscoveryRequestCallback
otThreadAnycastLocatorCallback
otNetDataGet
otNetDataGetLength
otNetDataGetMaxLength
otNetDataResetMaxLength
otNetDataGetNextOnMeshPrefix
otNetDataGetNextRoute
otNetDataGetNextService
otNetDataGetNextLowpanContextInfo
otNetDataGetCommissioningDataset
otNetDataGetVersion
otNetDataGetStableVersion
otNetDataSteeringDataCheckJoiner
otNetDataSteeringDataCheckJoinerWithDiscerner
otNetDataContainsOmrPrefix
otNetDataPublishDnsSrpServiceAnycast
otNetDataPublishDnsSrpServiceUnicast
otNetDataPublishDnsSrpServiceUnicastMeshLocalEid
otNetDataIsDnsSrpServiceAdded
otNetDataSetDnsSrpServicePublisherCallback
otNetDataUnpublishDnsSrpService
otNetDataPublishOnMeshPrefix
otNetDataPublishExternalRoute
otNetDataReplacePublishedExternalRoute
otNetDataIsPrefixAdded
otNetDataSetPrefixPublisherCallback
otNetDataUnpublishPrefix
otThreadGetNextDiagnosticTlv
otThreadSendDiagnosticGet
otThreadSendDiagnosticReset
otThreadGetVendorName
otThreadGetVendorModel
otThreadGetVendorSwVersion
otThreadSetVendorName
otThreadSetVendorModel
otThreadSetVendorSwVersion

otThreadSetEnabled
otThreadGetVersion
otThreadIsSingleton
otThreadDiscover
otThreadIsDiscoverInProgress
otThreadSetJoinerAdvertisement
otThreadGetChildTimeout
otThreadSetChildTimeout
otThreadGetExtendedPanId
otThreadSetExtendedPanId
otThreadGetLeaderRloc
otThreadGetLinkMode
otThreadSetLinkMode
otThreadGetNetworkKey
otThreadGetNetworkKeyRef
otThreadSetNetworkKey
otThreadSetNetworkKeyRef
otThreadGetRloc
otThreadGetMeshLocalEid
otThreadGetMeshLocalPrefix
otThreadSetMeshLocalPrefix
otThreadGetLinkLocalIp6Address
otThreadGetLinkLocalAllThreadNodesMulticastAddress
otThreadGetRealmLocalAllThreadNodesMulticastAddress
otThreadGetServiceAloc
otThreadGetNetworkName
otThreadSetNetworkName
otThreadGetDomainName
otThreadSetDomainName
otThreadSetFixedDuaInterfaceIdentifier
otThreadGetFixedDuaInterfaceIdentifier
otThreadGetKeySequenceCounter
otThreadSetKeySequenceCounter
otThreadGetKeySwitchGuardTime
otThreadSetKeySwitchGuardTime
otThreadBecomeDetached
otThreadBecomeChild
otThreadGetNextNeighborInfo
otThreadGetDeviceRole
otThreadDeviceRoleToString
otThreadGetLeaderData
otThreadGetLeaderRouterId
otThreadGetLeaderWeight
otThreadGetPartitionId

otThreadGetRloc16
otThreadGetParentInfo
otThreadGetParentAverageRssi
otThreadGetParentLastRssi
otThreadSearchForBetterParent
otThreadGetIp6Counters
otThreadResetIp6Counters
otThreadGetTimeInQueueHistogram
otThreadGetMaxTimeInQueue
otThreadResetTimeInQueueStat
otThreadGetMleCounters
otThreadResetMleCounters
otThreadRegisterParentResponseCallback
otThreadSetDiscoveryRequestCallback
otThreadLocateAnycastDestination
otThreadIsAnycastLocateInProgress
otThreadSendAddressNotification
otThreadSendProactiveBackboneNotification
otThreadDetachGracefully
otConvertDurationInSecondsToString
OT_NETWORK_DATA_ITERATOR_INIT
OT_SERVICE_DATA_MAX_SIZE
OT_SERVER_DATA_MAX_SIZE
OT_NETWORK_DIAGNOSTIC_TYPELIST_MAX_ENTRIES
OT_NETWORK_DIAGNOSTIC_CHILD_TABLE_ENTRY_SIZE
OT_NETWORK_DIAGNOSTIC_ITERATOR_INIT
OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_NAME_TLV_LENGTH
OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_MODEL_TLV_LENGTH
OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_SW_VERSION_TLV_LENGTH
OT_NETWORK_DIAGNOSTIC_MAX_THREAD_STACK_VERSION_TLV_LENGTH
OT_NETWORK_BASE_TLV_MAX_LENGTH
OT_NETWORK_MAX_ROUTER_ID
OT_NEIGHBOR_INFO_ITERATOR_INIT
OT_JOINER_ADVDATA_MAX_LENGTH
OT_DURATION_STRING_SIZE
Joiner
otJoinerDiscerner
mValue
mLength
otJoinerState
otJoinerState
otJoinerDiscerner
otJoinerCallback
otJoinerStart

otJoinerStop
otJoinerGetState
otJoinerGetId
otJoinerSetDiscerner
otJoinerGetDiscerner
otJoinerStateToString
OT_JOINER_MAX_DISCERNER_LENGTH
Operational Dataset
otNetworkKey
m8
otNetworkName
m8
otExtendedPanId
m8
otPskc
m8
otSecurityPolicy
mRotationTime
mObtainNetworkKeyEnabled
mNativeCommissioningEnabled
mRoutersEnabled
mExternalCommissioningEnabled
mCommercialCommissioningEnabled
mAutonomousEnrollmentEnabled
mNetworkKeyProvisioningEnabled
mTobleLinkEnabled
mNonCcmRoutersEnabled
mVersionThresholdForRouting
otOperationalDatasetComponents
mIsActiveTimestampPresent
mIsPendingTimestampPresent
mIsNetworkKeyPresent
mIsNetworkNamePresent
mIsExtendedPanIdPresent
mIsMeshLocalPrefixPresent
mIsDelayPresent
mIsPanIdPresent
mIsChannelPresent
mIsPskcPresent
mIsSecurityPolicyPresent
mIsChannelMaskPresent
otTimestamp
mSeconds
mTicks

mAuthoritative
otOperationalDataset
mActiveTimestamp
mPendingTimestamp
mNetworkKey
mNetworkName
mExtendedPanId
mMeshLocalPrefix
mDelay
mPanId
mChannel
mPskc
mSecurityPolicy
mChannelMask
mComponents
otOperationalDatasetTlvs
mTlvs
mLength
otMeshcopTlvType
otNetworkKey
otNetworkKeyRef
otNetworkName
otExtendedPanId
otMeshLocalPrefix
otPskc
otPskcRef
otSecurityPolicy
otChannelMask
otOperationalDatasetComponents
otTimestamp
otOperationalDataset
otOperationalDatasetTlvs
otMeshcopTlvType
otDatasetMgmtSetCallback
otDatasetUpdaterCallback
OT_TOOL_PACKED_END
otDatasetIsCommissioned
otDatasetGetActive
otDatasetGetActiveTlvs
otDatasetSetActive
otDatasetSetActiveTlvs
otDatasetGetPending
otDatasetGetPendingTlvs
otDatasetSetPending

otDatasetSetPendingTlvs
otDatasetSendMgmtActiveGet
otDatasetSendMgmtActiveSet
otDatasetSendMgmtPendingGet
otDatasetSendMgmtPendingSet
otDatasetGeneratePskc
otNetworkNameFromString
otDatasetParseTlvs
otDatasetConvertToTlvs
otDatasetUpdateTlvs
otDatasetCreateNewNetwork
otDatasetGetDelayTimerMinimal
otDatasetSetDelayTimerMinimal
otDatasetUpdaterRequestUpdate
otDatasetUpdaterCancelUpdate
otDatasetUpdaterIsUpdateOngoing
OT_NETWORK_KEY_SIZE
OT_NETWORK_NAME_MAX_SIZE
OT_EXT_PAN_ID_SIZE
OT_MESH_LOCAL_PREFIX_SIZE
OT_PSKC_MAX_SIZE
OT_CHANNEL_1_MASK
OT_CHANNEL_2_MASK
OT_CHANNEL_3_MASK
OT_CHANNEL_4_MASK
OT_CHANNEL_5_MASK
OT_CHANNEL_6_MASK
OT_CHANNEL_7_MASK
OT_CHANNEL_8_MASK
OT_CHANNEL_9_MASK
OT_CHANNEL_10_MASK
OT_CHANNEL_11_MASK
OT_CHANNEL_12_MASK
OT_CHANNEL_13_MASK
OT_CHANNEL_14_MASK
OT_CHANNEL_15_MASK
OT_CHANNEL_16_MASK
OT_CHANNEL_17_MASK
OT_CHANNEL_18_MASK
OT_CHANNEL_19_MASK
OT_CHANNEL_20_MASK
OT_CHANNEL_21_MASK
OT_CHANNEL_22_MASK
OT_CHANNEL_23_MASK

- OT_CHANNEL_24_MASK
- OT_CHANNEL_25_MASK
- OT_CHANNEL_26_MASK
- OT_OPERATIONAL_DATASET_MAX_LENGTH

Router/Leader

- otChildInfo
  - mExtAddress
  - mTimeout
  - mAge
  - mConnectionTime
  - mRloc16
  - mChildId
  - mNetworkDataVersion
  - mLinkQualityIn
  - mAverageRssi
  - mLastRssi
  - mFrameErrorRate
  - mMessageErrorRate
  - mQueuedMessageCnt
  - mSupervisionInterval
  - mVersion
  - mRxOnWhenIdle
  - mFullThreadDevice
  - mFullNetworkData
  - mIsStateRestoring
  - mIsCslSynced
- otCacheEntryInfo
  - mTarget
  - mRloc16
  - mState
  - mCanEvict
  - mRampDown
  - mValidLastTrans
  - mLastTransTime
  - mMeshLocalEid
  - mTimeout
  - mRetryDelay
- otCacheEntryIterator
  - mData
- otDeviceProperties
  - mPowerSupply
  - mIsBorderRouter
  - mSupportsCcm
  - mIsUnstable

mLeaderWeightAdjustment
otNeighborTableEntryInfo
mInstance
mChild
mRouter
mInfo
otCacheEntryState
otPowerSupply
otNeighborTableEvent
otChildIp6AddressIterator
otCacheEntryState
otCacheEntryInfo
otCacheEntryIterator
otDeviceProperties
otNeighborTableCallback
otThreadGetMaxAllowedChildren
otThreadSetMaxAllowedChildren
otThreadIsRouterEligible
otThreadSetRouterEligible
otThreadSetPreferredRouterId
otThreadGetDeviceProperties
otThreadSetDeviceProperties
otThreadGetLocalLeaderWeight
otThreadSetLocalLeaderWeight
otThreadGetPreferredLeaderPartitionId
otThreadSetPreferredLeaderPartitionId
otThreadGetJoinerUdpPort
otThreadSetJoinerUdpPort
otThreadSetSteeringData
otThreadGetContextIdReuseDelay
otThreadSetContextIdReuseDelay
otThreadGetNetworkIdTimeout
otThreadSetNetworkIdTimeout
otThreadGetRouterUpgradeThreshold
otThreadSetRouterUpgradeThreshold
otThreadGetChildRouterLinks
otThreadSetChildRouterLinks
otThreadReleaseRouterId
otThreadBecomeRouter
otThreadBecomeLeader
otThreadGetRouterDowngradeThreshold
otThreadSetRouterDowngradeThreshold
otThreadGetRouterSelectionJitter
otThreadSetRouterSelectionJitter

otThreadGetChildInfoById
otThreadGetChildInfoByIndex
otThreadGetChildNextIp6Address
otThreadGetRouterIdSequence
otThreadGetMaxRouterId
otThreadGetRouterInfo
otThreadGetNextCacheEntry
otThreadGetPskc
otThreadGetPskcRef
otThreadSetPskc
otThreadSetPskcRef
otThreadGetParentPriority
otThreadSetParentPriority
otThreadGetMaxChildIpAddresses
otThreadSetMaxChildIpAddresses
otThreadRegisterNeighborTableCallback
otThreadSetCcmEnabled
otThreadSetThreadVersionCheckEnabled
otThreadGetRouterIdRange
otThreadSetRouterIdRange
otThreadGetAdvertisementTrickleIntervalMax
otThreadIsRouterIdAllocated
otThreadGetNextHopAndPathCost
OT_CHILD_IP6_ADDRESS_ITERATOR_INIT
Server
otServerGetNetDataLocal
otServerAddService
otServerRemoveService
otServerGetNextService
otServerRegister
Add-Ons
Channel Manager
otChannelManagerRequestChannelChange
otChannelManagerGetRequestedChannel
otChannelManagerGetDelay
otChannelManagerSetDelay
otChannelManagerRequestChannelSelect
otChannelManagerSetAutoChannelSelectionEnabled
otChannelManagerGetAutoChannelSelectionEnabled
otChannelManagerSetAutoChannelSelectionInterval
otChannelManagerGetAutoChannelSelectionInterval
otChannelManagerGetSupportedChannels
otChannelManagerSetSupportedChannels
otChannelManagerGetFavoredChannels

- otChannelManagerSetFavoredChannels
- otChannelManagerGetCcaFailureRateThreshold
- otChannelManagerSetCcaFailureRateThreshold

Channel Monitoring

- otChannelMonitorSetEnabled
- otChannelMonitorIsEnabled
- otChannelMonitorGetSampleInterval
- otChannelMonitorGetRssiThreshold
- otChannelMonitorGetSampleWindow
- otChannelMonitorGetSampleCount
- otChannelMonitorGetChannelOccupancy

Child Supervision

- otChildSupervisionGetInterval
- otChildSupervisionSetInterval
- otChildSupervisionGetCheckTimeout
- otChildSupervisionSetCheckTimeout
- otChildSupervisionGetCheckFailureCounter
- otChildSupervisionResetCheckFailureCounter

CoAP

- CoAP
  - otCoapOption
    - mNumber
    - mLength
  - otCoapOptionIterator
    - mMessage
    - mOption
    - mNextOptionOffset
  - otCoapResource
    - mUriPath
    - mHandler
    - mContext
    - mNext
  - otCoapBlockwiseResource
    - mUriPath
    - mHandler
    - mReceiveHook
    - mTransmitHook
    - mContext
    - mNext
  - otCoapTxParameters
    - mAckTimeout
    - mAckRandomFactorNumerator
    - mAckRandomFactorDenominator
    - mMaxRetransmit

otCoapType
otCoapCode
otCoapOptionType
otCoapOptionContentFormat
otCoapBlockSzx
otCoapType
otCoapCode
otCoapOptionType
otCoapOption
otCoapOptionIterator
otCoapOptionContentFormat
otCoapBlockSzx
otCoapResponseHandler
otCoapRequestHandler
otCoapBlockwiseReceiveHook
otCoapBlockwiseTransmitHook
otCoapResource
otCoapBlockwiseResource
otCoapTxParameters
otCoapMessageInit
otCoapMessageInitResponse
otCoapMessageSetToken
otCoapMessageGenerateToken
otCoapMessageAppendContentFormatOption
otCoapMessageAppendOption
otCoapMessageAppendUintOption
otCoapMessageAppendObserveOption
otCoapMessageAppendUriPathOptions
otCoapBlockSizeFromExponent
otCoapMessageAppendBlock2Option
otCoapMessageAppendBlock1Option
otCoapMessageAppendProxyUriOption
otCoapMessageAppendMaxAgeOption
otCoapMessageAppendUriQueryOption
otCoapMessageSetPayloadMarker
otCoapMessageGetType
otCoapMessageGetCode
otCoapMessageSetCode
otCoapMessageCodeToString
otCoapMessageGetMessageId
otCoapMessageGetTokenLength
otCoapMessageGetToken
otCoapOptionIteratorInit
otCoapOptionIteratorGetFirstOptionMatching

otCoapOptionIteratorGetFirstOption
otCoapOptionIteratorGetNextOptionMatching
otCoapOptionIteratorGetNextOption
otCoapOptionIteratorGetOptionUintValue
otCoapOptionIteratorGetOptionValue
otCoapNewMessage
otCoapSendRequestWithParameters
otCoapSendRequestBlockWiseWithParameters
otCoapSendRequestBlockWise
otCoapSendRequest
otCoapStart
otCoapStop
otCoapAddResource
otCoapRemoveResource
otCoapAddBlockWiseResource
otCoapRemoveBlockWiseResource
otCoapSetDefaultHandler
otCoapSendResponseWithParameters
otCoapSendResponseBlockWiseWithParameters
otCoapSendResponseBlockWise
otCoapSendResponse
OT_DEFAULT_COAP_PORT
OT_COAP_DEFAULT_TOKEN_LENGTH
OT_COAP_MAX_TOKEN_LENGTH
OT_COAP_MAX_RETRANSMIT
OT_COAP_MIN_ACK_TIMEOUT
OT_COAP_CODE
CoAP Secure
otHandleCoapSecureClientConnect
otCoapSecureStart
otCoapSecureStop
otCoapSecureSetPsk
otCoapSecureGetPeerCertificateBase64
otCoapSecureSetSslAuthMode
otCoapSecureSetCertificate
otCoapSecureSetCaCertificateChain
otCoapSecureConnect
otCoapSecureDisconnect
otCoapSecureIsConnected
otCoapSecureIsConnectionActive
otCoapSecureSendRequestBlockWise
otCoapSecureSendRequest
otCoapSecureAddResource
otCoapSecureRemoveResource

otCoapSecureAddBlockWiseResource
otCoapSecureRemoveBlockWiseResource
otCoapSecureSetDefaultHandler
otCoapSecureSetClientConnectedCallback
otCoapSecureSendResponseBlockWise
otCoapSecureSendResponse
OT_DEFAULT_COAP_SECURE_PORT
Command Line Interface
otCliCommand
mName
mCommand
mCommand
otCliOutputCallback
otCliCommand
otCliInit
otCliInputLine
otCliSetUserCommands
otCliOutputBytes
otCliOutputFormat
otCliAppendResult
otCliPlatLogv
otCliVendorSetUserCommands
Crypto - Thread Stack
otCryptoSha256Hash
otCryptoHmacSha256
otCryptoAesCcm
Factory Diagnostics - Thread Stack
otDiagProcessCmd
otDiagProcessCmdLine
otDiagIsEnabled
Heap
otHeapCAlloc
otHeapFree
History Tracker
otHistoryTrackerIterator
mData32
mData16
otHistoryTrackerNetworkInfo
mRole
mMode
mRloc16
mPartitionId
otHistoryTrackerUnicastAddressInfo
mAddress

mPrefixLength
mAddressOrigin
mEvent
mScope
mPreferred
mValid
mRloc
otHistoryTrackerMulticastAddressInfo
mAddress
mAddressOrigin
mEvent
otHistoryTrackerMessageInfo
mPayloadLength
mNeighborRloc16
mSource
mDestination
mChecksum
mIpProto
mIcmp6Type
mAveRxRss
mLinkSecurity
mTxSuccess
mPriority
mRadioIeee802154
mRadioTrelUdp6
otHistoryTrackerNeighborInfo
mExtAddress
mRloc16
mAverageRssi
mEvent
mRxOnWhenIdle
mFullThreadDevice
mFullNetworkData
mIsChild
otHistoryTrackerRouterInfo
mEvent
mRouterId
mNextHop
mOldPathCost
mPathCost
otHistoryTrackerOnMeshPrefixInfo
mPrefix
mEvent
otHistoryTrackerExternalRouteInfo

mRoute
mEvent
otHistoryTrackerAddressEvent
@1
otHistoryTrackerNeighborEvent
otHistoryTrackerRouterEvent
otHistoryTrackerNetDataEvent
otHistoryTrackerIterator
otHistoryTrackerNetworkInfo
otHistoryTrackerUnicastAddressInfo
otHistoryTrackerMulticastAddressInfo
otHistoryTrackerMessageInfo
otHistoryTrackerNeighborInfo
otHistoryTrackerRouterInfo
otHistoryTrackerOnMeshPrefixInfo
otHistoryTrackerExternalRouteInfo
otHistoryTrackerInitIterator
otHistoryTrackerIterateNetInfoHistory
otHistoryTrackerIterateUnicastAddressHistory
otHistoryTrackerIterateMulticastAddressHistory
otHistoryTrackerIterateRxHistory
otHistoryTrackerIterateTxHistory
otHistoryTrackerIterateNeighborHistory
otHistoryTrackerIterateRouterHistory
otHistoryTrackerIterateOnMeshPrefixHistory
otHistoryTrackerIterateExternalRouteHistory
otHistoryTrackerEntryAgeToString
OT_HISTORY_TRACKER_MAX_AGE
OT_HISTORY_TRACKER_ENTRY_AGE_STRING_SIZE
OT_HISTORY_TRACKER_NO_NEXT_HOP
OT_HISTORY_TRACKER_INFINITE_PATH_COST
Jam Detection
otJamDetectionCallback
otJamDetectionSetRssiThreshold
otJamDetectionGetRssiThreshold
otJamDetectionSetWindow
otJamDetectionGetWindow
otJamDetectionSetBusyPeriod
otJamDetectionGetBusyPeriod
otJamDetectionStart
otJamDetectionStop
otJamDetectionIsEnabled
otJamDetectionGetState

otJamDetectionGetHistoryBitmap
Logging - Thread Stack
otLogHexDumpInfo
mDataBytes
mDataLength
mTitle
mLine
mIterator
otLoggingGetLevel
otLoggingSetLevel
otLogCritPlat
otLogWarnPlat
otLogNotePlat
otLogInfoPlat
otLogDebgPlat
otDumpCritPlat
otDumpWarnPlat
otDumpNotePlat
otDumpInfoPlat
otDumpDebgPlat
otLogPlat
otLogPlatArgs
otLogCli
otLogGenerateNextHexDumpLine
OT_LOG_HEX_DUMP_LINE_SIZE
Mesh Diagnostics
otMeshDiagDiscoverConfig
mDiscoverIp6Addresses
mDiscoverChildTable
otMeshDiagRouterInfo
mExtAddress
mRloc16
mRouterId
mVersion
mIsThisDevice
mIsThisDeviceParent
mIsLeader
mIsBorderRouter
mLinkQualities
mIp6AddrIterator
mChildIterator
otMeshDiagChildInfo
mRloc16
mMode

mLinkQuality
mIsThisDevice
mIsBorderRouter
otMeshDiagChildEntry
mRxOnWhenIdle
mDeviceTypeFtd
mFullNetData
mCslSynchronized
mSupportsErrRate
mRloc16
mExtAddress
mVersion
mTimeout
mAge
mConnectionTime
mSupervisionInterval
mLinkMargin
mAverageRssi
mLastRssi
mFrameErrorRate
mMessageErrorRate
mQueuedMessageCount
mCslPeriod
mCslTimeout
mCslChannel
otMeshDiagRouterNeighborEntry
mSupportsErrRate
mRloc16
mExtAddress
mVersion
mConnectionTime
mLinkMargin
mAverageRssi
mLastRssi
mFrameErrorRate
mMessageErrorRate
otMeshDiagDiscoverConfig
otMeshDiagIp6AddrIterator
otMeshDiagChildIterator
otMeshDiagRouterInfo
otMeshDiagChildInfo
otMeshDiagDiscoverCallback
otMeshDiagChildEntry
otMeshDiagQueryChildTableCallback

- otMeshDiagChildIp6AddrsCallback
- otMeshDiagRouterNeighborEntry
- otMeshDiagQueryRouterNeighborTableCallback
- otMeshDiagDiscoverTopology
- otMeshDiagCancel
- otMeshDiagGetNextIp6Address
- otMeshDiagGetNextChildInfo
- otMeshDiagQueryChildTable
- otMeshDiagQueryChildrenIp6Addrs
- otMeshDiagQueryRouterNeighborTable
- OT_MESH_DIAG_VERSION_UNKNOWN

Network Co-Processor

- otNcpHdlcSendCallback
- otNcpDelegateAllowPeekPoke
- otNcpHdlcSendDone
- otNcpHdlcReceive
- otNcpHdlcInit
- otNcpSpiInit
- otNcpStreamWrite
- otNcpPlatLogv
- otNcpRegisterPeekPokeDelegates

Network Time Synchronization

- otNetworkTimeStatus
- otNetworkTimeStatus
- otNetworkTimeSyncCallbackFn
- otNetworkTimeGet
- otNetworkTimeSetSyncPeriod
- otNetworkTimeGetSyncPeriod
- otNetworkTimeSetXtalThreshold
- otNetworkTimeGetXtalThreshold
- otNetworkTimeSyncSetCallback
- OT_TIME_SYNC_INVALID_SEQ

Radio Statistics

- otRadioTimeStats
  - mDisabledTime
  - mSleepTime
  - mTxTime
  - mRxTime
- otRadioTimeStats
- otRadioTimeStatsGet
- otRadioTimeStatsReset

Random Number Generator

- RNG Cryptographic
  - otRandomCryptoFillBuffer

- RNG Non-cryptographic
  - otRandomNonCryptoGetUint32
  - otRandomNonCryptoGetUint8
  - otRandomNonCryptoGetUint16
  - otRandomNonCryptoGetUint8InRange
  - otRandomNonCryptoGetUint16InRange
  - otRandomNonCryptoGetUint32InRange
  - otRandomNonCryptoFillBuffer
  - otRandomNonCryptoAddJitter
- SNTP
  - otSntpQuery
    - mMessageInfo
  - otSntpQuery
  - otSntpResponseHandler
  - otSntpClientQuery
  - otSntpClientSetUnixEra
  - OT_SNTP_DEFAULT_SERVER_IP
  - OT_SNTP_DEFAULT_SERVER_PORT

Platform Abstraction

- Alarm
  - otPlatAlarmMicroStartAt
  - otPlatAlarmMicroStop
  - otPlatAlarmMicroGetNow
  - otPlatAlarmMicroFired
  - otPlatAlarmMilliStartAt
  - otPlatAlarmMilliStop
  - otPlatAlarmMilliGetNow
  - otPlatAlarmMilliFired
  - otPlatDiagAlarmFired
- Crypto - Platform
  - otCryptoKey
    - mKey
    - mKeyLength
    - mKeyRef
  - otCryptoContext
    - mContext
    - mContextSize
  - otPlatCryptoSha256Hash
    - m8
  - otPlatCryptoEcdsaKeyPair
    - mDerBytes
    - mDerLength
  - otPlatCryptoEcdsaPublicKey
    - m8

otPlatCryptoEcdsaSignature
m8
otCryptoKeyType
otCryptoKeyAlgorithm
@11
otCryptoKeyStorage
otCryptoKeyRef
otCryptoKey
otCryptoContext
otPlatCryptoSha256Hash
otPlatCryptoEcdsaKeyPair
otPlatCryptoEcdsaPublicKey
otPlatCryptoEcdsaSignature
OT_TOOL_PACKED_END
otPlatCryptoInit
otPlatCryptoImportKey
otPlatCryptoExportKey
otPlatCryptoDestroyKey
otPlatCryptoHasKey
otPlatCryptoHmacSha256Init
otPlatCryptoHmacSha256Deinit
otPlatCryptoHmacSha256Start
otPlatCryptoHmacSha256Update
otPlatCryptoHmacSha256Finish
otPlatCryptoAesInit
otPlatCryptoAesSetKey
otPlatCryptoAesEncrypt
otPlatCryptoAesFree
otPlatCryptoHkdfInit
otPlatCryptoHkdfExpand
otPlatCryptoHkdfExtract
otPlatCryptoHkdfDeinit
otPlatCryptoSha256Init
otPlatCryptoSha256Deinit
otPlatCryptoSha256Start
otPlatCryptoSha256Update
otPlatCryptoSha256Finish
otPlatCryptoRandomInit
otPlatCryptoRandomDeinit
otPlatCryptoRandomGet
otPlatCryptoEcdsaGenerateKey
otPlatCryptoEcdsaGetPublicKey
otPlatCryptoEcdsaSign
otPlatCryptoEcdsaVerify

otPlatCryptoEcdsaSignUsingKeyRef
otPlatCryptoEcdsaExportPublicKey
otPlatCryptoEcdsaGenerateAndImportKey
otPlatCryptoEcdsaVerifyUsingKeyRef
otPlatCryptoPbkdf2GenerateKey
OT_CRYPTO_SHA256_HASH_SIZE
OT_CRYPTO_ECDSA_MAX_DER_SIZE
OT_CRYPTO_ECDSA_PUBLIC_KEY_SIZE
OT_CRYPTO_ECDSA_SIGNATURE_SIZE
OT_CRYPTO_PBDKF2_MAX_SALT_SIZE
DNS - Platform
otPlatDnsUpstreamQuery
otPlatDnsStartUpstreamQuery
otPlatDnsCancelUpstreamQuery
otPlatDnsUpstreamQueryDone
Entropy
otPlatEntropyGet
Factory Diagnostics - Platform
otGpioMode
otPlatDiagProcess
otPlatDiagModeSet
otPlatDiagModeGet
otPlatDiagChannelSet
otPlatDiagTxPowerSet
otPlatDiagRadioReceived
otPlatDiagAlarmCallback
otPlatDiagGpioSet
otPlatDiagGpioGet
otPlatDiagGpioSetMode
otPlatDiagGpioGetMode
otPlatDiagRadioSetRawPowerSetting
otPlatDiagRadioGetRawPowerSetting
otPlatDiagRadioRawPowerSettingEnable
otPlatDiagRadioTransmitCarrier
otPlatDiagRadioTransmitStream
otPlatDiagRadioGetPowerSettings
Logging - Platform
otLogRegion
otLogLevel
otLogRegion
otPlatLog
otPlatLogHandleLevelChanged
OT_LOG_LEVEL_NONE
OT_LOG_LEVEL_CRIT

- OT_LOG_LEVEL_WARN
- OT_LOG_LEVEL_NOTE
- OT_LOG_LEVEL_INFO
- OT_LOG_LEVEL_DEBG

Memory
- otPlatCAlloc
- otPlatFree

Message Pool
- otMessageBuffer
  - mNext
- otMessageBuffer
- otPlatMessagePoolInit
- otPlatMessagePoolNew
- otPlatMessagePoolFree
- otPlatMessagePoolNumFreeBuffers

Miscellaneous
- otPlatResetReason
- otPlatMcuPowerState
- otPlatReset
- otPlatResetToBootloader
- otPlatGetResetReason
- otPlatAssertFail
- otPlatWakeHost
- otPlatSetMcuPowerState
- otPlatGetMcuPowerState

Network Simulator
- otPlatOtnsStatus

Radio
- Radio Types
  - otExtAddress
    - m8
  - otMacKey
    - m8
  - otMacKeyMaterial
    - mKeyRef
    - mKey
    - mKeyMaterial
  - otRadioIeInfo
    - mNetworkTimeOffset
    - mTimeIeOffset
    - mTimeSyncSeq
  - otRadioFrame
    - mPsdu
    - mLength

mChannel
mRadioType
mIid
mAesKey
mIeInfo
mTxDelayBaseTime
mTxDelay
mMaxCsmaBackoffs
mMaxFrameRetries
mRxChannelAfterTxDone
mIsHeaderUpdated
mIsARetx
mCsmaCaEnabled
mCslPresent
mIsSecurityProcessed
mTxInfo
mTimestamp
mAckFrameCounter
mAckKeyId
mRssi
mLqi
mAckedWithFramePending
mAckedWithSecEnhAck
mRxInfo
mInfo
otRadioCoexMetrics
mNumGrantGlitch
mNumTxRequest
mNumTxGrantImmediate
mNumTxGrantWait
mNumTxGrantWaitActivated
mNumTxGrantWaitTimeout
mNumTxGrantDeactivatedDuringRequest
mNumTxDelayedGrant
mAvgTxRequestToGrantTime
mNumRxRequest
mNumRxGrantImmediate
mNumRxGrantWait
mNumRxGrantWaitActivated
mNumRxGrantWaitTimeout
mNumRxGrantDeactivatedDuringRequest
mNumRxDelayedGrant
mAvgRxRequestToGrantTime
mNumRxGrantNone

mStopped
otLinkMetrics
mPduCount
mLqi
mLinkMargin
mRssi
mReserved
@12
@13
@14
@15
@16
otRadioKeyType
otRadioState
otRadioCaps
otPanId
otShortAddress
otExtAddress
otMacKey
otMacKeyRef
otMacKeyMaterial
otRadioIeInfo
otRadioFrame
otRadioState
otRadioCoexMetrics
otLinkMetrics
OT_TOOL_PACKED_END
OT_PANID_BROADCAST
OT_EXT_ADDRESS_SIZE
CSL_IE_HEADER_BYTES_LO
CSL_IE_HEADER_BYTES_HI
OT_MAC_KEY_SIZE
OT_TOOL_PACKED_END
Radio Configuration
otPlatRadioGetCaps
otPlatRadioGetVersionString
otPlatRadioGetReceiveSensitivity
otPlatRadioGetIeeeEui64
otPlatRadioSetPanId
otPlatRadioSetExtendedAddress
otPlatRadioSetShortAddress
otPlatRadioGetTransmitPower
otPlatRadioSetTransmitPower
otPlatRadioGetCcaEnergyDetectThreshold

otPlatRadioSetCcaEnergyDetectThreshold
otPlatRadioGetFemLnaGain
otPlatRadioSetFemLnaGain
otPlatRadioGetPromiscuous
otPlatRadioSetPromiscuous
otPlatRadioSetMacKey
otPlatRadioSetMacFrameCounter
otPlatRadioSetMacFrameCounterIfLarger
otPlatRadioGetNow
otPlatRadioGetBusSpeed
Radio Operation
otPlatRadioGetState
otPlatRadioEnable
otPlatRadioDisable
otPlatRadioIsEnabled
otPlatRadioSleep
otPlatRadioReceive
otPlatRadioReceiveAt
otPlatRadioReceiveDone
otPlatDiagRadioReceiveDone
otPlatRadioGetTransmitBuffer
otPlatRadioTransmit
otPlatRadioTxStarted
otPlatRadioTxDone
otPlatDiagRadioTransmitDone
otPlatRadioGetRssi
otPlatRadioEnergyScan
otPlatRadioEnergyScanDone
otPlatRadioEnableSrcMatch
otPlatRadioAddSrcMatchShortEntry
otPlatRadioAddSrcMatchExtEntry
otPlatRadioClearSrcMatchShortEntry
otPlatRadioClearSrcMatchExtEntry
otPlatRadioClearSrcMatchShortEntries
otPlatRadioClearSrcMatchExtEntries
otPlatRadioGetSupportedChannelMask
otPlatRadioGetPreferredChannelMask
otPlatRadioSetCoexEnabled
otPlatRadioIsCoexEnabled
otPlatRadioGetCoexMetrics
otPlatRadioEnableCsl
otPlatRadioUpdateCslSampleTime
otPlatRadioGetCslAccuracy
otPlatRadioGetCslUncertainty

otPlatRadioSetChannelMaxTransmitPower
otPlatRadioSetRegion
otPlatRadioGetRegion
otPlatRadioConfigureEnhAckProbing
otPlatRadioAddCalibratedPower
otPlatRadioClearCalibratedPowers
otPlatRadioSetChannelTargetPower
otPlatRadioGetRawPowerSetting
Radio Extension
otPlatRadioExtensionCoexEvent_t
otPlatRadioExtensionGetTxAntennaMode
otPlatRadioExtensionSetTxAntennaMode
otPlatRadioExtensionGetRxAntennaMode
otPlatRadioExtensionSetRxAntennaMode
otPlatRadioExtensionGetActivePhy
otPlatRadioExtensionGetDpState
otPlatRadioExtensionSetDpState
otPlatRadioExtensionGetGpioInputOverride
otPlatRadioExtensionSetGpioInputOverride
otPlatRadioExtensionGetActiveRadio
otPlatRadioExtensionGetPhySelectTimeout
otPlatRadioExtensionSetPhySelectTimeout
otPlatRadioExtensionGetCoexOptions
otPlatRadioExtensionSetCoexOptions
otPlatRadioExtensionGetCoexConstantOptions
otPlatRadioExtensionIsCoexEnabled
otPlatRadioExtensionSetCoexEnable
otPlatRadioExtensionGetRequestPwmArgs
otPlatRadioExtensionSetRequestPwmArgs
otPlatRadioExtensionClearCoexCounters
otPlatRadioExtensionGetCoexCounters
otPlatRadioExtensionSetRadioHoldoff
otPlatRadioExtensionGetRadioCounters
otPlatRadioExtensionClearRadioCounters
Settings
@21
otPlatSettingsInit
otPlatSettingsDeinit
otPlatSettingsGet
otPlatSettingsSet
otPlatSettingsAdd
otPlatSettingsDelete
otPlatSettingsWipe
SPI Slave

- otPlatSpiSlaveTransactionCompleteCallback
- otPlatSpiSlaveTransactionProcessCallback
- otPlatSpiSlaveEnable
- otPlatSpiSlaveDisable
- otPlatSpiSlavePrepareTransaction

Time Service
- otPlatTimeGet
- otPlatTimeGetXtalAccuracy

Toolchain
- OT_MUST_USE_RESULT
- OT_TOOL_PACKED_BEGIN
- OT_TOOL_PACKED_FIELD
- OT_TOOL_WEAK
- OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK
- OT_UNUSED_VARIABLE
- OT_UNREACHABLE_CODE
- OT_APPLE_IGNORE_GNU_FOLDING_CONSTANT
- OT_FALL_THROUGH

TREL - Platform
- otPlatTrelPeerInfo
  - mRemoved
  - mTxtData
  - mTxtLength
  - mSockAddr
- otPlatTrelPeerInfo
- otPlatTrelEnable
- otPlatTrelDisable
- otPlatTrelHandleDiscoveredPeerInfo
- otPlatTrelRegisterService
- otPlatTrelSend
- otPlatTrelHandleReceived

Infrastructure Interface
- otPlatInfraIfHasAddress
- otPlatInfraIfSendIcmp6Nd
- otPlatInfraIfRecvIcmp6Nd
- otPlatInfraIfStateChanged
- otPlatInfraIfDiscoverNat64Prefix
- otPlatInfraIfDiscoverNat64PrefixDone

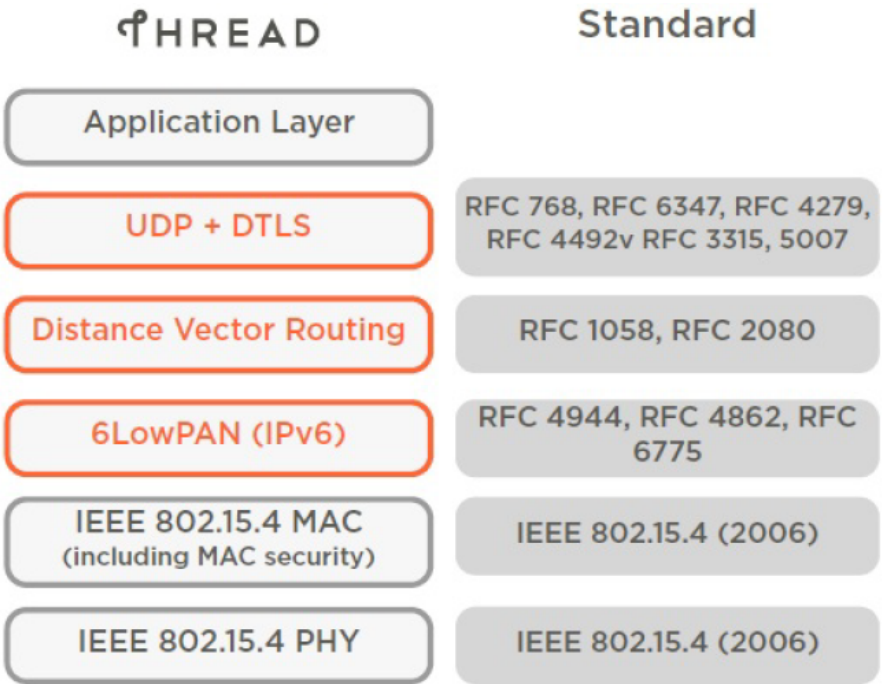
OpenThread Modules

Developing with OpenThread

Developing Silicon Labs OpenThread Applications

OpenThread released by Google is:

- **An open-source implementation of the Thread® networking protocol.** Google Nest has released OpenThread to make the technology used in Nest products more broadly available to developers to accelerate the development of products for the connected home.
- **OS and platform agnostic**, with a narrow platform abstraction layer and a small memory footprint, making it highly portable. It supports both system-on-chip (SoC) and co-processor designs.
- **A Thread Certified Component**, implementing all features defined in the Thread 1.3.0 specification, including all Thread networking layers (IPv6, 6LoWPAN, IEEE 802.15.4 with MAC security, Mesh Link Establishment, Mesh Routing) and device roles, as well as Border Router support.



Silicon Labs has implemented an OpenThread-based protocol tailored to work with Silicon Labs hardware. This protocol is available on GitHub and also as a software development kit (SDK) installed with Simplicity Studio 5. The SDK is a fully tested snapshot of the GitHub source. It supports a broader range of hardware than does the GitHub version, and includes documentation and example applications not available on GitHub.The content on these pages is intended for those who want to experiment with or are already developing an OpenThread application using the Silicon Labs OpenThread SDK.

For details about this release: Links to release notes are available on the silabs.com Gecko SDK page.

For Silicon Labs' OpenThread product information: See the product pages on silabs.com.

For background about the OpenThread protocol and other wireless networking topics: The Fundamentals section is a good place to start. This section also contains information and resources for the open source OpenThread protocol.

To get started with development: See the Getting Started section to get started working with example applications.

If you are already in development: See the Developer's Guide for details or go directly to the API Reference.

Overview

Getting Started with Silicon Labs OpenThread Development

To get started with Silicon Labs OpenThread development, download the Simplicity Studio Development environment as described in the Simplicity Studio 5 User's Guide.

This will also prompt you to install the Gecko SDK (GSDK), which contains the OpenThread SDK. The GSDK combines Silicon Labs wireless software development kits (SDKs) and Gecko Platform into a single, integrated package. The GSDK is your primary tool for developing in the Silicon Labs IoT Software ecosystem. All of Silicon Labs' stacks are written in-house to provide a seamless experience from silicon to tools, allowing you to unlock powerful features with ease, including:

- Abstraction of complex requirements like multiprotocol and pre-certification
- Industry-leading ability to support a large number of nodes
- Ultra-low power consumption
- Strong network reliability

Silicon Labs also helps future-proof your devices with over-the-air software and security updates, helping to minimize maintenance cost and improve your end user product experience!

Once you have downloaded Simplicity Studio and the GSDK, detailed instructions for using the OpenThread examples and configuration tools are provided in the **OpenThread SDK Quick-Start Guide (PDF)**.

Note: The recommended method to get started with the GSDK is to first install Simplicity Studio 5, which will set up your development environment and walk you through the GSDK installation. Alternatively, GSDK and other required tools may be installed manually from the GitHub GSDK site.

Overview

OpenThread Fundamentals

Silicon Labs has produced a series of documents on topics that provide useful background for Silicon Labs OpenThread developers. This page also includes resources for OpenThread.

Silicon Labs OpenThread Resources

- Thread Fundamentals: For those new to OpenThread, includes a brief background on the emergence of Thread, provides a technology overview, and describes some key features of Thread to consider when implementing a Thread solution.
- Wireless Networking Fundamentals: For those new to wireless networking, introduces some fundamental concepts of wireless networking.

Additional OpenThread Resources

Documentation

Generic OpenThread end-user documentation and guides are located at openthread.io. Here you can:

- Learn more about OpenThread features and enhancements
- Use OpenThread in your products
- Learn how to build and configure a Thread network
- Port OpenThread to a new platform
- Build an application on top of OpenThread -Certify a product using OpenThread

Note: For users in China, end-user documentation is available at openthread.google.cn.

Contributing

We would love for you to contribute to OpenThread and help make it even better than it is today! See the Contributing Guidelines for more information.

Contributors are required to abide by OpenThread Code of Conduct and Coding Conventions and Style Guide.

Versioning

OpenThread follows the Semantic Versioning guidelines for release cycle transparency and to maintain backwards compatibility. OpenThread's versioning is independent of the Thread protocol specification version but will clearly indicate which version of the specification it currently supports.

License

OpenThread is released under the BSD 3-Clause license. See the LICENSE file for more information.

Only use the OpenThread name and marks when accurately referencing this software distribution. Do not use the marks in a way that suggests you are endorsed by or otherwise affiliated with Nest, Google, or The Thread Group.

Need help?

There are numerous avenues for OpenThread support:

- Bugs and feature requests — submit to the Issue Tracker
- Stack Overflow — post questions using the openthread tag
- Google Groups — discussion and announcements at openthread-users This is the recommended place for users to discuss OpenThread and interact directly with the OpenThread team.

Overview

OpenThread Developers Guide

The Developer's Guide content is organized in the following groups:

- Developing and Debugging: A description of development resources as well as detailed information on a variety of topics.
- OpenThread Border Router: About developing and using an OpenThread Border Router
- Coexistence: Background on coexistence issues and strategies for improving performance in the presence of other protocol traffic.
- Multiprotocol: Background on implementing multiprotocol applications and information on different multiprotocol models.
- Bootloading: Information on using the Gecko Bootloader with OpenThread applications.
- Non-Volatile Memory: Background on managing device memory.
- Security: Describes Silicon Labs security resources and how to manage OpenThread security.
- Performance: Provides performance testing and measurement tools and techniques as well as results.

Overview

Developing and Debugging Silicon Labs OpenThread Applications

These pages provide details on developing and debugging OpenThread applications. Content includes:

- **Configuring and Building OpenThread Applications for Sleepy Devices**: Describes how to configure OpenThread applications to operate on a proprietary sub-GHz band using the Silicon Labs OpenThread SDK and Simplicity Studio 5 with a compatible mainboard. It also provides details on the proprietary radio PHY supported with this feature.
- **Single-Band Proprietary Sub-GHz Support with OpenThread**: Describes how to configure OpenThread applications to operate on a proprietary sub-GHz band using the Silicon Labs OpenThread SDK and Simplicity Studio 5 with a compatible mainboard. It also provides details on the proprietary radio PHY supported with this feature.
- **Using OpenThread with Free RTOS**: Describes how to build OpenThread applications with FreeRTOS.
- **Configuring OpenThread Applications for Thread 1.3**: Provides instructions for configuring OpenThread SoC and Border Router applications to use Thread 1.3 features.

Development Tools

Simplicity Studio and the Simplicity IDE: Simplicity Studio® is the unified development environment for all Silicon Labs technologies, SoCs, and modules. It provides you with access to the target device-specific web and SDK resources, software and hardware configuration tools, and an integrated development environment (IDE) featuring industry-standard code editors, compilers, and debuggers. See the silabs.com Simplicity Studio page to download the tools and for more information.

Network Analyzer: Simplicity Studio® 5 (SSv5)'s Network Analyzer enables debugging of complex wireless systems. This tool captures a trace of wireless network activity that can be examined in detail live or at a later time. See the Network Analyzer section of the Simplicity Studio 5 User's Guide for more information.

Wireshark: Download instructions are provided for Windows/Mac users or Linux users. Simplicity Studio® 5 supports live interaction between the application running on a Silicon Labs device and Wireshark.

Energy Profiler: Simplicity Studio® 5 (SSv5)'s Energy Profiler enables you to visualize the energy consumption of individual devices, multiple devices on one target system, or a network of interacting wireless devices to analyze and improve the power performance of these systems. Real-time information on current consumption is correlated with the program counter providing advanced energy software monitoring capabilities. It also provides a basic level of integration with the Network Analyzer network analysis tool. See the Energy Profiler section of the Simplicity Studio 5 User's Guide for more information.

Simplicity Commander: Simplicity Commander is a single, all-purpose tool to be used in a production environment. It is invoked using a simple Command Line Interface (CLI) that is also scriptable. Simplicity Commander enables customers to complete essential tasks such as configuring and building applications and bootloaders and flashing images to their devices. Simplicity Commander is available through Simplicity Studio or can be downloaded through system-specific installers. The **Simplicity Commander User's Guide** provides more information.

Silicon Labs Configurator (SLC): SLC offers command-line access to application configuration and generation functions. **Software Project Generation and Configuration with SLC-CLI** provides instructions on downloading and using the SLC-CLI tool.

Configuring Sleepy Devices

Configuring and Building OpenThread Applications for Sleepy Devices

Minimal Thread Devices (MTDs) communicate only through their Thread Router parent and cannot relay messages for other devices. **Sleepy Devices** are MTDs that achieve low-power duty cycling by turning off their transceiver to reduce power. For more information on these device roles in a Thread network, refer to the Thread specification or UG103.11: Thread Fundamentals.

There are two categories of Sleepy Devices:

- Regular Sleepy End Devices (SEDs) that poll for new messages when they wake up.
- Synchronized Sleepy End Devices (SSEDs) that use IEEE 802.15.4 Coordinated Sampled Listening (CSL) so that the parent can forward its messages during designated transmission and reception windows.

Silicon Labs provides sample applications to demonstrate both kinds of Sleepy End Device behavior.

Sleepy Devices can greatly help extend the battery power of power-limited devices. Note that simply building a Sleepy Device application does not guarantee the lowest power consumption, as this is dependent on many configuration parameters as well as the application's general use case. Additionally, any application code such as shell or CLI, LCD code, or other peripheral components can also adversely affect the power consumption. For an example of such current consumption optimization, refer to the Silicon Labs example at https://github.com/SiliconLabs/zigbee_applications/tree/master/zigbee_sed_z3switch.

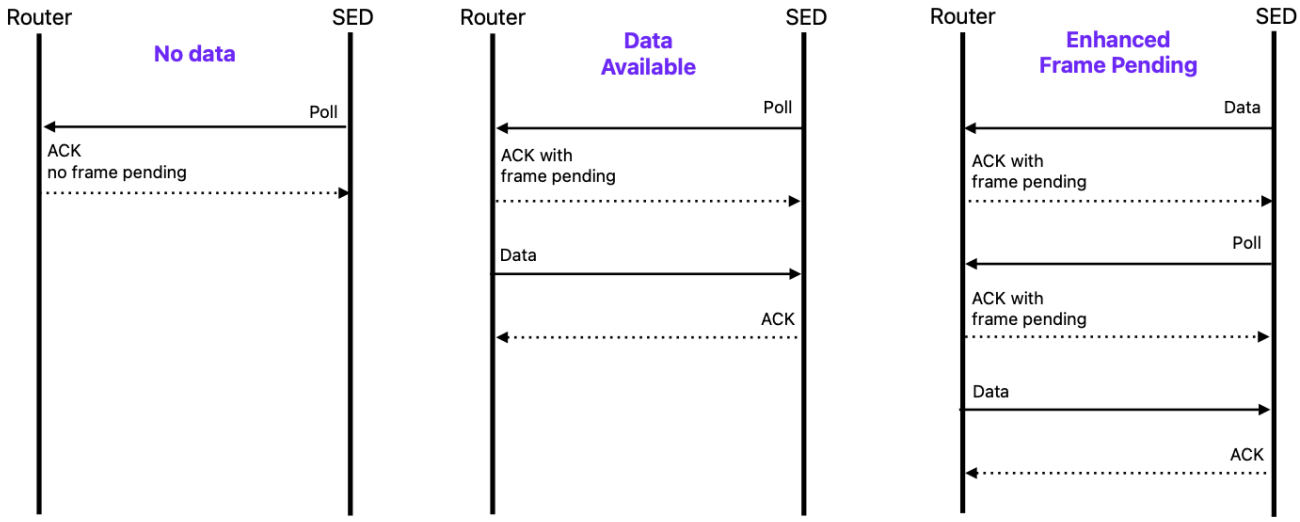
Note: This document describes how to configure Sleepy Devices. Further detail about power consumption, optimizing current consumption values, or providing benchmark values for various platforms is outside the document's scope.

Sleepy End Devices (SED)

Sleepy End Devices (SEDs)

SEDs achieve lower power consumption by sleeping for a set period, periodically waking up to send data polls (MAC data requests) to their parent. If the parent has any pending data to send to its child, it is indicated by a frame pending bit in the 802.15.4 Acknowledgement to the data poll. This lets the Sleepy End Device keep its receiver on for the anticipated data which the parent will send immediately.

Starting with Thread 1.2, the OpenThread stack automatically also makes use of the 802.15.4 Enhanced Frame Pending feature, which lets the SED use regular data messages to get an indication of pending data. Without this feature, the SED would have to send a data poll on its scheduled period to extract the data from the parent.



Note that a smaller poll period (that is, polling more frequently) means better latency at the cost of higher power consumption.

SED Configuration in OpenThread

APIs:

- `otLinkGetPollPeriod` / `otLinkSetPollPeriod` : Get/Set/Clear user-specified/external data poll period for sleepy end device.

Warning: The following poll-related configuration items have standardized values in the Thread specification. Changing them might affect the certifiability of your component or end product.

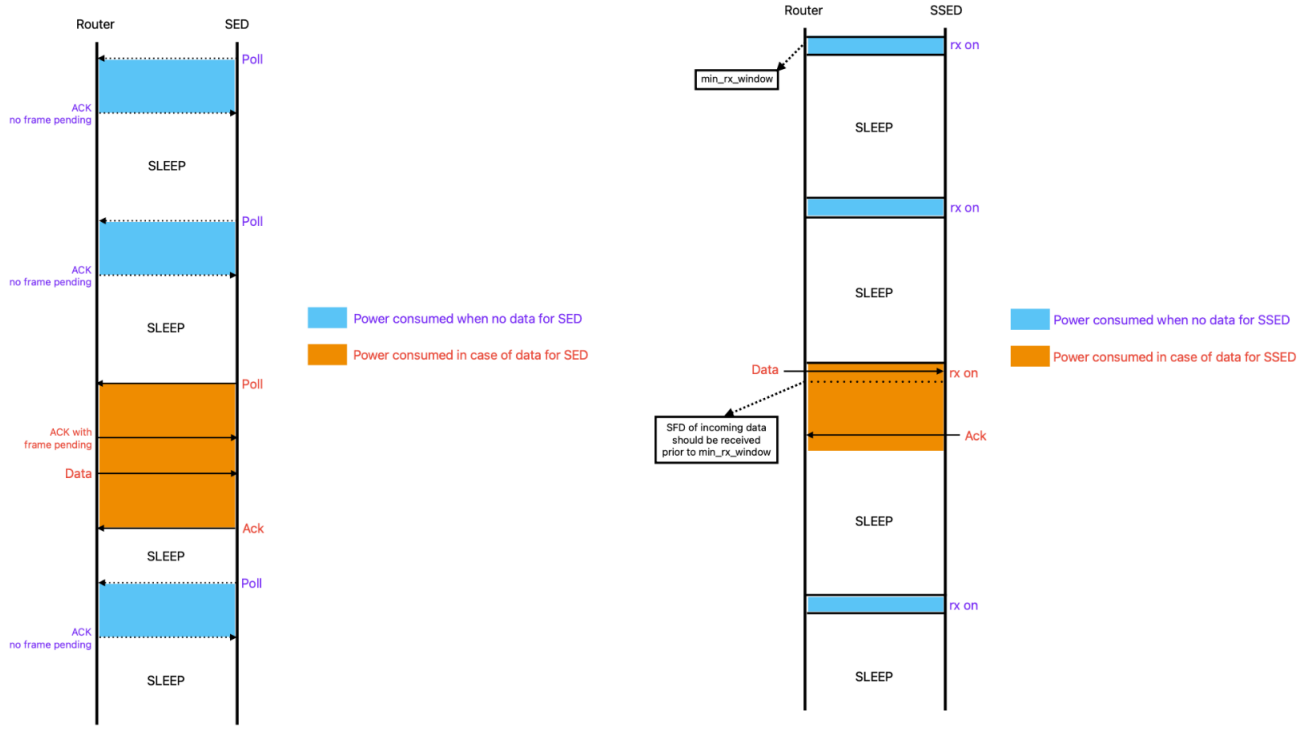
- OPENTHREAD_CONFIG_MAC_MAX_TX_ATTEMPTS_INDIRECT_POLLS: Maximum number of received IEEE 802.15.4 Data Requests for a queued indirect transaction.
- OPENTHREAD_CONFIG_MAC_ATTACH_DATA_POLL_PERIOD: The Data Poll period during attach in milliseconds.
- OPENTHREAD_CONFIG_MAC_MINIMUM_POLL_PERIOD: Minimum poll period in milliseconds.
- OPENTHREAD_CONFIG_MAC_RETX_POLL_PERIOD: Retransmission poll period in milliseconds.

Synchronized Sleepy End Devices (SSED)

Synchronized Sleepy End Devices (SSEDs)

An SSED is an rx-off-when-idle end device that uses the IEEE 802.15.4-2015 CSL feature, available beginning with Thread 1.2, to further optimize power consumption. An SSED CSL receiver is synchronized with a parent that is a CSL transmitter, so both parent and child require implementation of this feature.

Coordinated Sampled Listening (CSL) involves the receiver sampling for any data from the transmitter during set intervals. The transmitter always targets the synchronized window, eliminating the need for data polling and optimizing power consumption. In the following figure, note the comparison of activity with a regular SED.



SED Power Consumption

SSED Power Consumption

SSEDs automatically reduce the average TX-ON time by avoiding wasted data polling. Therefore, optimized power consumption depends on configuring minimal average RX-ON time for a platform and the given application use case.

Note that an SSED periodically must send some packets within a specified timeout to maintain synchronization with its parent. In OpenThread, the auto-synchronization happens automatically using data polls (usually at a much less frequent rate than regular polling). Note that other data packets from the SSED, including application data, can also be used by the stack to re-synchronize the connection as needed.

IEEE 802.15.4-2015 Coordinated Sampled Listening (CSL)

CSL Parameters and CSL Information Elements

Following are the parameters that indicate CSL configuration on a receiver:

- **CSL Period**: A CSL receiver performs periodic channel sampling by configuring a non-zero period value *(see macCslPeriod in [IEEE802154-2015])*.
- **CSL Phase**: Thread uses the CSL phase definition from *[IEEE802154-2015]*: "*the time from the first symbol of the frame containing the CSL IE ... until the next channel sample*".
  - "First symbol" is interpreted as the first symbol of the MAC header.
  - The CSL receiver should be ready to receive slightly earlier than the preamble time (CSL-Phase – CCA time) and should stay in receive mode until after the CSL-Phase time to detect the Sync-Frame-Detect (SFD) of the incoming packet from the CSL transmitter. These timing values account for the platform's implementation and accuracy.
- **CSL Channel**: If the CSL receiver expects to receive and process unsynchronized CSL transmissions, then it should use a different channel from the Thread network channel for receiving CSL messages. However, most Thread applications use cases for SSEDs only involve synchronized communications, so the CSL channel can remain the same as the network channel.
- **CSL Timeout**: Timeout before which an SSED and its parent must re-synchronize to keep the connection valid and active.

CSL synchronization happens by the SSED child communicating CSL parameters to its parent. The CSL channel and timeout are initially configured during mesh link establishment (MLE) attach. The period and phase are communicated by the receiver in IEEE 802.15.4 Information Elements in the MAC header. The **CSL IE** in the IEEE 802.15.4 MAC header is a tuple containing **[CSL period, CSL phase]**.

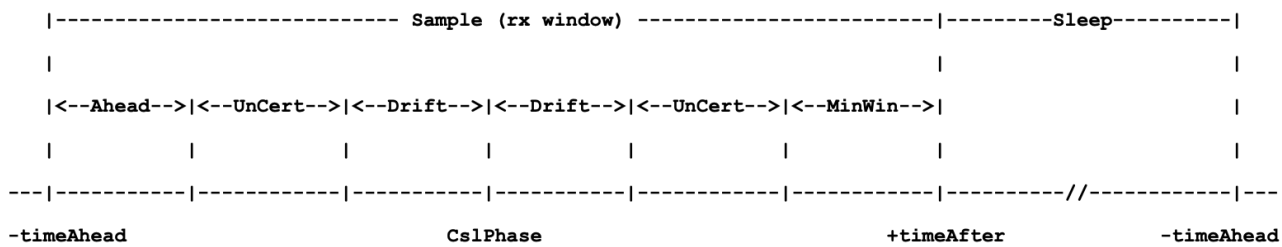
An SSED device should include CSL IEs in all the IEEE 802.15.4 Commands, ACKs, and Data frames unicast to its parent router. IEEE 802.15.4 ACKs that include IEs are called **Enhanced Acknowledgements** (EnhAcks), as defined in *[IEEE802154-2015]*.

Note that the CSL period, channel, and timeout are configured by the application, whereas the phase value is dynamically determined on the SSED relative to the exact moment the frame containing the CSL IE is sent out to the parent. CSL retransmissions involve recalculating the CSL IE with a new phase value.

OpenThread CSL Timing Calculations

The CSL transmitter's delay for its scheduled transmission points to the moment when the end of the SFD will be present at the receiver's local antenna, relative to the local radio clock. The CSL receiver should be ready to receive the first symbol of a scheduled frame's Sync Header (SHR) at its own receive window start time. If no SHR is detected at the end of its minimum receive window, the radio should be turned off or switched to TX mode as needed.

The CSL sample window of the CSL receiver extends before and after its calculated sample time. This marks a timestamp in the CSL sample window where a frame would be received in "ideal conditions" if there was no inaccuracy or clock drift. However, the realistic sampling representation is as follows:



- **timeAhead** accounts for the SSED to wake and be ready for RX.
- **timeAfter** is the when the RX-ON window closes. If needed, the Silicon Labs radio driver automatically extends the duration of the receive window.
- **Uncert** is the fixed uncertainty (that is, random jitter) of the arrival time of CSL transmissions. In addition to uncertainty accumulated over elapsed time, the CSL channel sample ("RX window") must be extended by twice this value such that an actual transmission is guaranteed to be detected by the local receiver in the presence of random arrival time jitter.
- The calculations also account for clock drift and the estimated worst-case accuracy (maximum ± deviation from the nominal frequency) of the local radio clock used to schedule CSL operations.

SSED Configuration in OpenThread

APIs:

- `otPlatRadioEnableCsl` : Enable or disable CSL receiver.
- `otPlatRadioUpdateCslSampleTime` : Update CSL sample time in radio driver.
- `otPlatRadioGetCslAccuracy` : Get the current estimated worst-case accuracy of the local radio clock in PPM.
- `otPlatRadioGetCslUncertainty` : The fixed uncertainty (that is, random jitter) of the arrival time of CSL transmissions received by this device in 10-microsecond units.
- `otLinkGetCslChannel` / `otLinkSetCslChannel` : Get/Set CSL Channel.
- `otLinkGetCslPeriod` / `otLinkSetCslPeriod` : Get/Set CSL Period.
- `otLinkGetCslTimeout` / `otLinkSetCslTimeout` : Get/Set CSL Timeout.
- `otLinkIsCslEnabled` : Indicates whether or not CSL is enabled.
- `otLinkIsCslSupported` : Indicates whether the device is connected to a parent that supports CSL.

Configurable parameters:

The following parameters are configurable in the OpenThread stack component in Simplicity Studio or at run-time using the Silicon Labs Configurator (SLC).

- OPENTHREAD_CONFIG_MAC_CSL_RECEIVER_ENABLE: Configure CSL receiver support at build time.
- SL_OPENTHREAD_CSL_TX_UNCERTAINTY: CSL Scheduling Uncertainty (±10 microseconds). SSED's receive window will increase by twice this value.
- OPENTHREAD_CONFIG_MAC_CSL_DEBUG_ENABLE: Enable CSL debug printing (will affect timing, so use only for debug).

WARNING: The following configuration parameters have standardized values for Silicon Labs platforms and changing them might affect certifiability of your component or end product.

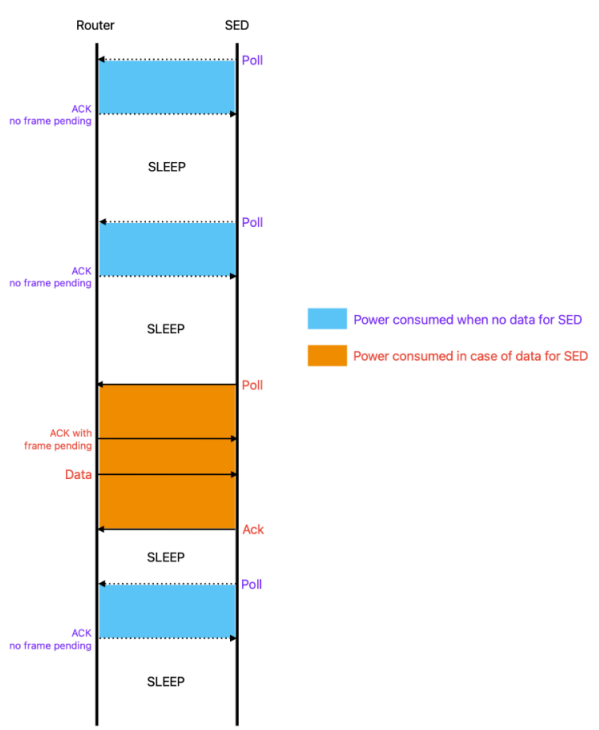
- OPENTHREAD_CONFIG_MAC_CSL_TRANSMITTER_ENABLE: Enable CSL transmitter functions. Automatically enabled for Thread 1.2 or greater devices.
- OPENTHREAD_CONFIG_MAC_CSL_AUTO_SYNC_ENABLE: Configure CSL auto synchronization based on data poll mechanism in Thread 1.2. This is turned off for some reference devices for certification testing purposes. For OpenThread device end products, this should never be turned off.
- OPENTHREAD_CONFIG_MAC_CSL_MIN_PERIOD: Minimum CSL period in milliseconds.
- OPENTHREAD_CONFIG_MAC_CSL_MAX_TIMEOUT: Maximum CSL timeout in seconds.
- OPENTHREAD_CONFIG_CSL_TIMEOUT: Default CSL timeout in seconds.
- OPENTHREAD_CONFIG_MAC_CSL_REQUEST_AHEAD_US: For a CSL transmitter, this indicates the time, measured in microseconds, by which the MAC should advance the delivery of the CSL frame to the radio layer before the actual transmit time.
- OPENTHREAD_CONFIG_CSL_TRANSMIT_TIME_AHEAD: Transmission scheduling and ramp-up time needed for the CSL transmitter to be ready, in microseconds.
- OPENTHREAD_CONFIG_CSL_RECEIVE_TIME_AHEAD: Reception scheduling and ramp up time needed for the CSL receiver to be ready, in f microseconds.
- OPENTHREAD_CONFIG_MIN_RECEIVE_ON_AHEAD: The minimum time (in microseconds) before the MAC Header (MHR) start that the radio should be in the receive state in order to properly receive any IEEE 802.15.4 frame. Defaults to the duration of Sync Header (SHR) + PHY Header (PHR).
- OPENTHREAD_CONFIG_MIN_RECEIVE_ON_AFTER: The minimum time (in microseconds) after the MHR start that the radio should be in receive state in order to properly receive any IEEE 802.15.4 frame. For Silicon Labs products, this value is zero, as the Silicon Labs radio driver will automatically extend the receive window when the SHR is detected.
- SL_OPENTHREAD_HFXO_ACCURACY: Worst case HFXO XTAL accuracy in units of ± ppm. Set to platform's SL_DEVICE_INIT_HFXO_PRECISION value.
- SL_OPENTHREAD_LFXO_ACCURACY: Worst case LFXO XTAL accuracy in units of ± ppm. Set to platform's SL_DEVICE_INIT_LFXO_PRECISION value.

SSED Use Cases

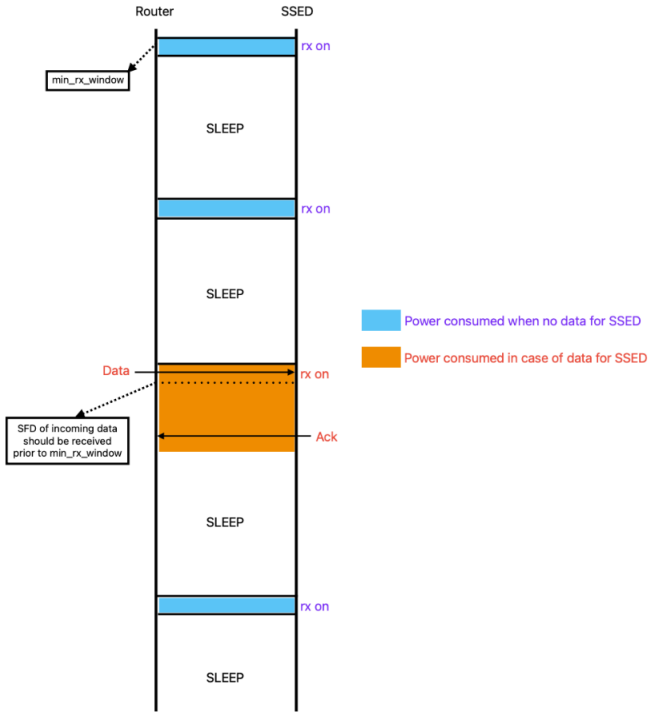
SSED Use Cases

- In dense network settings, SSEDs can reduce over-the-air radio traffic by avoiding frequent data polling.
- SSEDs are also useful in settings with sparse data traffic patterns that require high responsivity from the sleepy devices.

When comparing viability of SEDs and SSEDs, note that SSEDs may have lower power consumption in a direct comparison only in settings that ensure tight CSL accuracy and uncertainty. With worse uncertainty deviations, or in settings with high interference, any power savings sought by reducing time spent in data polling traffic will be compromised by power spent on larger receive windows. Referring to the following figure, the goal should be to minimize receive windows on the SSEDs when there is no data, while still supporting the application use case.



SED Power Consumption



SSED Power Consumption

Building And Using Silicon Labs Sleepy End Device Demo Applications

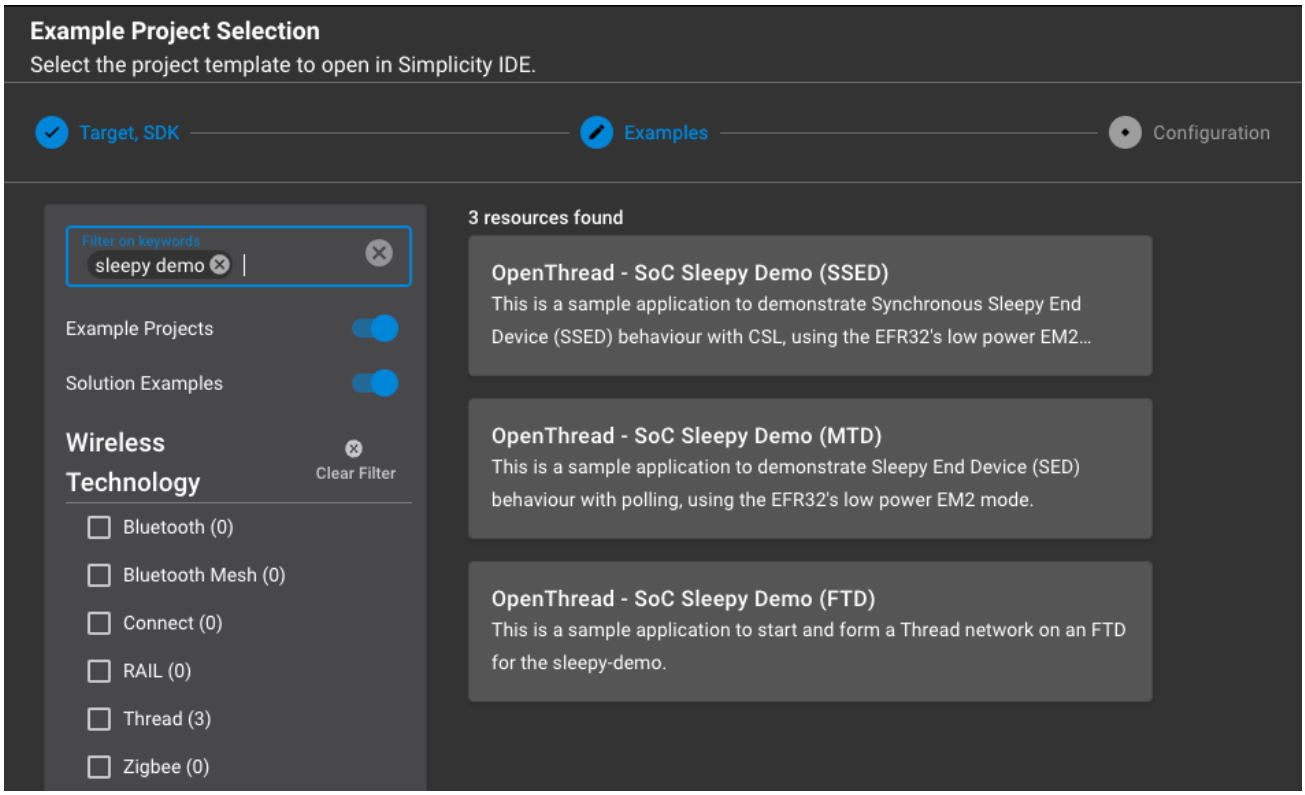
Building and Using Silicon Labs Sleepy End Device Demo Applications

The EFR32 Sleepy applications demonstrate Sleepy End Device behavior using the EFR32's low power deep sleep EM2 mode.

- **OpenThread – SoC Sleepy Demo (FTD) (sleepy-demo-ftd):** An application to start and form a Thread network on a Full Thread Device (FTD) for the sleepy-demo. This application is used in conjunction with the other sleepy demo applications.
- **OpenThread – SoC Sleepy Demo (sleepy-demo-mtd):** An application to demonstrate Sleepy End Device (SED) behavior on a Minimal Thread Device (MTD) that attaches to a Thread network started by a node running **sleepy-demo-ftd**. This application demonstrates power manager feature support and EM2 mode for the EFR32.
- **OpenThread – SoC Synchronized Sleepy Demo (sleepy-demo-ssed):** An application to demonstrate SSED behavior using CSL that attaches to a Thread network started by a node running **sleepy-demo-ftd**. This application demonstrates power manager feature support and EM2 mode for the EFR32.

Building Sleepy Demo Applications

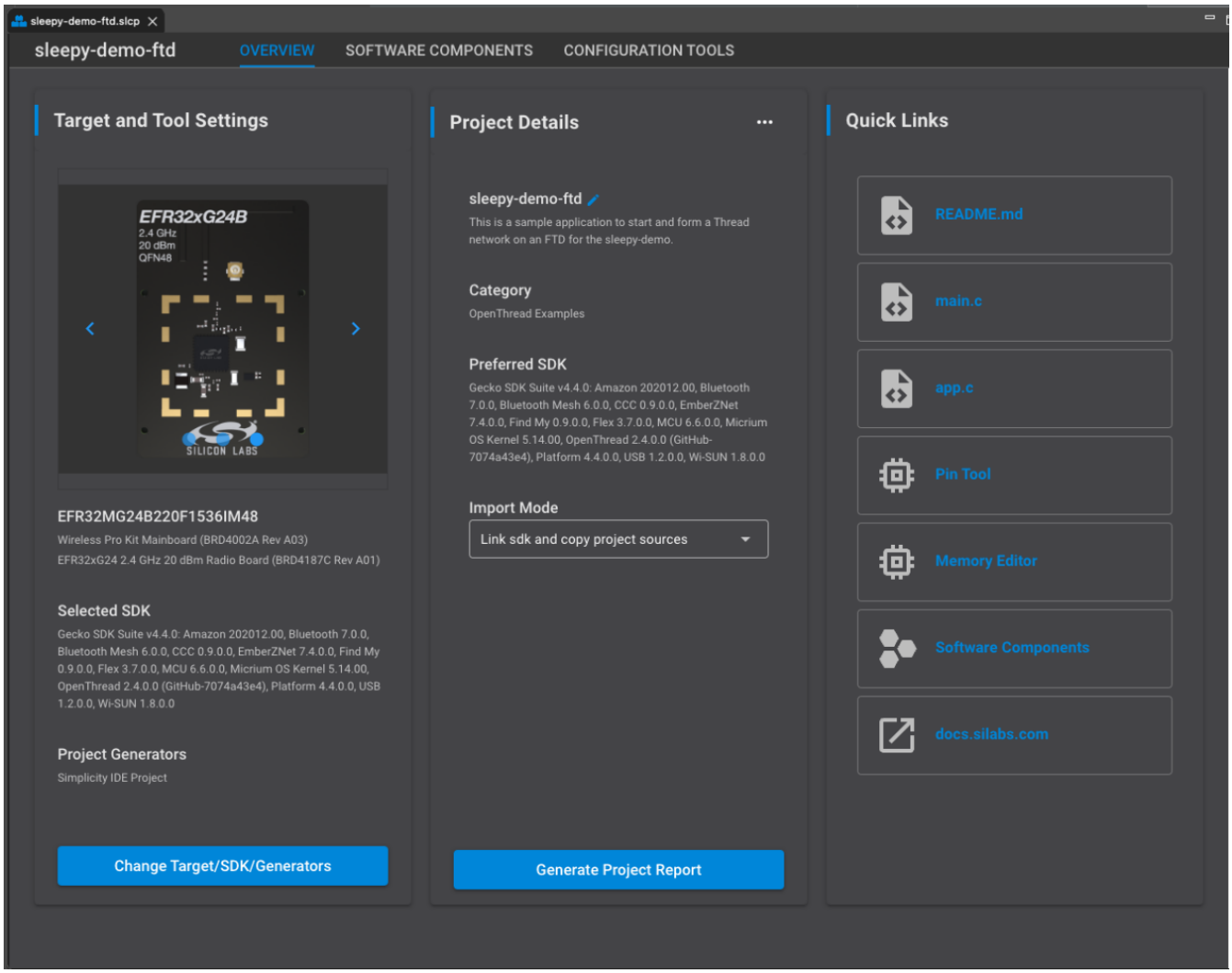
1. With the target part connected to your computer, open Simplicity Studio 5's **File** menu and select **New > Silicon Labs Project Wizard**.
2. The Target, SDK, and Toolchain Selection dialog opens. Click **NEXT**.
3. The Example Project Selection dialog opens. Use the Technology Type and Keyword filters to search for "**sleepy demo**" as the keyword.



- To build an FTD that can act as a parent / router for the example, select **sleepy-demo-ftd**.
- To demonstrate a regular Sleepy End Device that can work with the **sleepy-demo-ftd** as its parent, select **sleepy-demo-mtd**.
- To demonstrate a Synchronized Sleepy End Device that can use CSL with the **sleepy-demo-ftd** as its parent, select **sleepy-demo-ssed**.

The Project Configuration dialog opens.

4. Rename the project, change the default project file location, and determine if you will link to or copy project files. Note that, if you change any linked resource, it is changed for any other project that references it. Click **FINISH**.
5. The Simplicity IDE Perspective opens with the Project Configurator open to the **OVERVIEW** tab. See the online Simplicity Studio 5 User's Guide for details about the functionality available through the Simplicity IDE perspective and the Project Configurator. The following is an example of how the **sleepy-demo-ftd** project will look in this perspective.



6. Make any configuration changes to the software components. The autogeneration progress is available in the bottom right of the Simplicity IDE perspective. Make sure that progress is complete before you build.
7. Compile and flash the application image as described in QSG170: Silicon Labs OpenThread Quick Start Guide .

Demonstration

For demonstration purposes the network settings are hardcoded within the source files. Within a few seconds of powering on, the devices start Thread and form a network. In a real-life application the devices should implement and go through a commissioning process to create a network and add devices.

When the **sleepy-demo-ftd** device is started, the CLI displays:

```
sleepy-demo-ftd started
sleepy-demo-ftd changed to leader
```

When the **sleepy-demo-mtd** device is started, the CLI displays:

```
sleepy-demo-mtd started
[poll period: 2000 ms.]
```

The application is configured to join the pre-configured Thread network, disabling Rx-on-when-idle mode to become a Sleepy End Device. The default poll period is set in sleepy-mtd.c.

Issue the command to retrieve child table in the FTD console and observe that the R (Rx-on-when-idle) flag of the child is 0.

```
> child table
| ID  | RLOC16 | Timeout    | Age        | LQ In | C_VN |R|D|N|Ver|CSL|QMsgCnt|Suprvsn| Extended MAC     |
+-----+--------+------------+------------+-------+------+-+-+-+---+---+-------+-------+------------------+
|   1 | 0x8401 |        240 |          3 |     3 |    3 |0|0|0|  4| 0 |     0 |   129 | 667bf54fcc2aed8a |
Done
```

When the **sleepy-demo-ssed** device is started, the CLI displays:

```
sleepy-demo-ssed started
[csl period: 500000 us.] [csl timeout: 20 sec.]
```

The application is configured to join the pre-configured Thread network, disabling Rx-on-when-idle mode to become a Synchronous Sleepy End Device. The default CSL parameters are set in sleepy-ssed.c

Issue the command to retrieve the child table in the FTD console and observe that the R (Rx-on-when-idle) flag of the child is 0 and that the CSL flag is 1.

```
> child table
| ID  | RLOC16 | Timeout    | Age        | LQ In | C_VN |R|D|N|Ver|CSL|QMsgCnt|Suprvsn| Extended MAC     |
+-----+--------+------------+------------+-------+------+-+-+-+---+---+-------+-------+------------------+
|   1 | 0x8402 |        240 |          3 |     3 |    3 |0|0|0|  4| 1 |     0 |   129 | 8e8582dbd78c243c |
Done
```

Buttons on the MTD/SSED

Pressing button 0 on the MTD/SSED toggles between EM2 (sleep) and EM1 (idle) modes.

Pressing button 1 on the MTD/SSED sends a multicast UDP message containing a pre-defined string. The FTD listens on the multicast address and displays “Message Received: <string>” in the CLI.

Buttons on the FTD

Pressing either button 0 or 1 on the FTD sends a UDP message to the FTD containing the string "ftd button". Before pressing either button on the FTD, press the MTD's or SSED's button 1 to send a multicast message so that the FTD knows the address of the sleepy device to send messages to.

Monitoring Power Consumption of the MTD/SSED

Open the Energy Profiler in Simplicity Studio 5 (SSv5). In the Quick Access menu select ***Start Energy Capture...*** and select the MTD / SSED device. For further information on monitoring power consumption and energy profiler, see sections 5.3 and 5.3.1 in QSG170: Silicon Labs OpenThread QuickStart Guide.

Notes on Sleeping, Sleepy Callback, and Interrupts

To allow the EFR32 to enter sleepy mode, the application must register a callback with efr32SetSleepCallback. The return value of the callback is used to indicate that the application has no further work to do and that it is safe to go into a low

power mode. Because the callback is called with interrupts disabled, it should do the minimum required to check if it can sleep.

Overview

OpenThread Border Router

A Thread Border Router connects a Thread Network to other IPbased networks, such as Wi-Fi® or Ethernet®. A Thread Network requires a Border Router to connect to other networks. The Border Router provides services for devices within the Thread Network, including routing services for off-network operations, bidirectional connectivity over IPv6 infrastructure links, and service registry to enable DNS-based service discovery.

- Using the Silicon Labs RCP with the OpenThread Border Router: Describes using the OpenThread Border Router GitHub repository and the Silicon Labs OpenThread Radio Co-Processor (RCP) application to create a Thread Border Router.

Overview

Coexistence

This section describes implementing managed coexistence to improve coexistence of 2.4 GHz IEEE 802.11b/g/n Wi-Fi and other 2.4 GHz radios with IEEE 802.15.4-based radios such as Zigbee® and OpenThread.

- **Wi-Fi Coexistence Fundamentals**: Introduces methods to improve the coexistence of 2.4 GHz IEEE 802.11b/g/n Wi-Fi and other 2.4 GHz radios such as Bluetooth, Bluetooth Mesh, Bluetooth Low Energy, and IEEE 802.15.4-based radios such as Zigbee and OpenThread.
- **Zigbee and OpenThread Coexistence with Wi-Fi**: Details the impact of Wi-Fi on Zigbee and Thread, and methods to improve coexistence. First, methods to improve coexistence without direct interaction between Zigbee/Thread and Wi-Fi radios are described. Second, Silicon Labs's Packet Traffic Arbitration (PTA) support to coordinate 2.5 GHz RF traffic for co-located Zigbee/Thread and Wi-Fi radios is described (for the EFR32MG only).
- **Configuring Antenna Diversity for OpenThread**: Describes how to use Project Configurator and Component Editor in Simplicity Studio 5 to configure antenna diversity in OpenThread applications.

Overview

OpenThread in Multiprotocol Applications

This section provides background information on multiprotocol applications, and details on using OpenThread in multiprotocol applications, including dynamic multiprotocol and concurrent multiprotocol models.

- Multiprotocol Fundamentals (PDF): Describes the four multiprotocol modes, discusses considerations when selecting protocols for multiprotocol implementations, and reviews the Radio Scheduler, a required component of a dynamic multiprotocol solution.
- Dynamic Multiprotocol User's Guide (PDF): Describes how to implement a dynamic multiprotocol solution.
- Dynamic Multiprotocol Development with Bluetooth and OpenThread on SoCs (PDF): Provides details on developing Dynamic Multiprotocol applications for SoCs using Bluetooth and OpenThread.
- Running Zigbee, OpenThread, and Bluetooth Concurrently on a Linux Host with a Multiprotocol Co-Processor (PDF): Describes how to run any combination of Zigbee EmberZNet, OpenThread, and Bluetooth networking stacks on a Linux host processor, interfacing with a single EFR32 Radio Coprocessor (RCP) with multiprotocol and multi-PAN support. It also describes how to run the Zigbee stack on the EFR32 as a network co-processor (NCP) alongside the OpenThread RCP.
- Running Zigbee, OpenThread, and Bluetooth Concurrently on a System-on-Chip (PDF): Describes how to run a combination of Zigbee, Bluetooth, and OpenThread networking stacks and the Zigbee application layer on a System-on-Chip (SoC).
- Using the Co-Processor Communication Daemon (CPCd) (PDF): Documents the steps needed to properly configure and run the CPC daemon (CPCd) on Linux or Android.

Overview

Bootloading Embedded Applications

Bootloading allows you to update application firmware images on your devices. This section provides background information about bootloading using the Silicon Labs Gecko Bootloader.

- Bootloader Fundamentals: Introduces bootloading for Silicon Labs networking devices. Discusses the Gecko Bootloader as well as legacy Ember and Bluetooth bootloaders, and describes the file formats used by each.
- Gecko Bootloader User's Guide: Describes the high-level implementation of the Silicon Labs Gecko Bootloader for EFR32 SoCs and NCPs, and provides information on how to get started using the Gecko Bootloader with Silicon Labs wireless protocol stacks in GSDK 4.0 and higher.
- Series 2 Secure Boot with RTSL: Contains detailed information on configuring and using the Secure Boot with hardware Root of Trust and Secure Loader on Series 2 devices, including how to provision the signing key. This is a companion document to UG266: Silicon Labs Gecko Bootloader User's Guide.
- Transitioning to the Updated Gecko Bootloader in GSDK 4.0 and Higher: Gecko Bootloader v2.x, introduced in GSDK 4.0, contains a number of changes compared to Gecko Bootloader v1.x. This document describes the differences between the versions, including how to configure the new Gecko Bootloader in Simplicity Studio 5.

Overview

Non-Volatile Data Storage

This section offers an introduction to non-volatile data storage and describes how to use NVM3 data storage.

- Non-Volatile Data Storage Fundamentals: Introduces non-volatile data storage using flash and the three different storage implementations offered for Silicon Labs microcontrollers and SoCs: Simulated EEPROM, PS Store, and NVM3.
- Using NVM3 Data Storage: Explains how NVM3 can be used as non-volatile data storage in various protocol implementations.

Overview

Security

Silicon Labs offers a range of security features depending on the part you are using and your application and production needs. As well as the security features available, this section describes security issues specific to OpenThread.

- IoT Security Fundamentals: Introduces the security concepts that must be considered when implementing an Internet of Things (IoT) system. Using the ioXt Alliance's eight security principles as a structure, it clearly delineates the solutions Silicon Labs provides to support endpoint security and what you must do outside of the Silicon Labs framework.
- Using Silicon Labs Secure Vault Features with OpenThread: Describes how Secure Vault features are leveraged in OpenThread applications. It focuses on specific PSA features and emphasizes how these are integrated into the OpenThread stack.
- Series 2 Secure Debug: Describes how to lock and unlock the debug access of EFR32 Gecko Series 2 devices. Many aspects of the debug access, including the secure debug unlock are described. The Debug Challenge Interface (DCI) and Secure Engine (SE) Mailbox Interface for locking and unlocking debug access are also included.
- Production Programming of Series 2 Devices: Provides details on programming, provisioning, and configuring Series 2 devices in production environments. Covers Secure Engine Subsystem of Series 2 devices, which runs easily upgradeable Secure Engine (SE) or Virtual Secure Engine (VSE) firmware.
- Anti-Tamper Protection Configuration and Use: Shows how to program, provision, and configure the anti-tamper module on EFR32 Series 2 devices with Secure Vault.
- Authenticating Silicon Labs Devices using Device Certificates: Shows how to authenticate an EFR32 Series 2 device with Secure Vault, using secure device certificates and signatures.
- Secure Key Storage: Explains how to securely "wrap" keys in EFR32 Series 2 devices with Secure Vault, so they can be stored in non-volatile storage.
- Programming Series 2 Devices Using the DCI and SWD: Describes how to provision and configure Series 2 devices through the DCI and SWD.
- Integrating Crypto Functionality with PSA Crypto vs. Mbed TLS: Describes how to integrate crypto functionality into applications using PSA Crypto compared to Mbed TLS.
- Series 2 TrustZone (PDF): Provides background and information on implementing TrustZone on series 2 devices.

Overview

Performance

This section describes tools to use to help improve performance as well as network performance results.

- Manufacturing Test Overview: Provides a high-level description of the different options for integrating RF testing and characterization into standard test flows. It is intended for customers who are moving from the early prototype development stage to the manufacturing production environment and need assistance with manufacturing test.
- Manufacturing Test Guidelines for the EFR32: Details the different options for integrating RF testing and characterization into standard test flows.
- Performance Results for Multi-PAN RCP for OpenThread and Zigbee: Summarizes the performance test effort and results for some testing scenarios of the CPCd interface using multi-PAN for both OpenThread and Zigbee protocols.
- Mesh Network Performance Comparison: Reviews the Zigbee, Thread, and Bluetooth mesh networks to evaluate their differences in performance and behavior.
- Thread Mesh Network Performance (requires login to the customer portal): Details methods for testing Thread mesh network performance; results are intended to provide guidance on design practices and principles as well as expected field performance results.

OpenThread Modules

OpenThread Modules

Modules	
Alarm	This module includes the platform abstraction for the alarm service.
Backbone Router	This module includes functions for the OpenThread Backbone Router Service.
Border Agent	This module includes functions for the Thread Border Agent role.
Border Router	This module includes functions to manage local network data with the OpenThread Border Router.
Border Routing Manager	This module includes definitions related to Border Routing Manager.
Channel Manager	This module includes functions for Channel Manager.
Channel Monitoring	This module includes functions for channel monitoring feature.
Child Supervision	This module includes functions for Child Supervision feature.
CoAP	This module includes functions that control CoAP communication.
CoAP Secure	This module includes functions that control CoAP Secure (CoAP over DTLS) communication.
Command Line Interface	This module includes functions that control the Thread stack's execution.
Commissioner	This module includes functions for the Thread Commissioner role.
Crypto - Platform	This module includes the platform abstraction for Crypto.
Crypto - Thread Stack	This module includes cryptographic functions.
DNS	This module includes functions that control DNS communication.
DNS - Platform	This module includes the platform abstraction for sending recursive DNS query to upstream DNS servers.
DNS-SD Server	This module includes APIs for DNS-SD server.
Entropy	This module includes the platform abstraction for entropy generation.
Error	This module includes error definitions used in OpenThread.
Factory Diagnostics - Platform	This module includes the platform abstraction for diagnostics features.
Factory Diagnostics - Thread Stack	This module includes functions that control the Thread stack's execution.
General	This module includes functions for all Thread roles.
Heap	This module includes functions that set the external OpenThread heap.
History Tracker	Records the history of different events, for example RX and TX messages or network info changes.
ICMPv6	This module includes functions that control ICMPv6 communication.
IPv6	This module includes functions that control IPv6 communication.
Infrastructure Interface	This module includes the platform abstraction for the adjacent infrastructure network interface.
Instance	This module includes functions that control the OpenThread Instance.

Modules	
Jam Detection	This module includes functions for signal jamming detection feature.
Joiner	This module includes functions for the Thread Joiner role.
Link	This module includes functions that control link-layer configuration.
Link Metrics	This module includes functions that control the Link Metrics protocol.
Logging - Platform	This module includes the platform abstraction for the debug log service.
Logging - Thread Stack	This module includes OpenThread logging related definitions.
Memory	This module includes the platform abstraction for dynamic memory allocation.
Mesh Diagnostics	This module includes definitions and functions for Mesh Diagnostics.
Message	This module includes functions that manipulate OpenThread message buffers.
Message Pool	This module includes the platform abstraction for the message pool.
Miscellaneous	This module includes platform abstractions for miscellaneous behaviors.
Multi Radio Link	This module includes definitions and functions for multi radio link.
NAT64	This module includes functions and structs for the NAT64 function on the border router.
Network Co-Processor	This module includes functions that control the Thread stack's execution.
Network Simulator	This module includes the platform abstraction for OTNS.
Network Time Synchronization	This module includes functions that control network time synchronization service.
Operational Dataset	Includes functions for the Operational Dataset API.
Ping Sender	This file includes the OpenThread API for the ping sender module.
RNG Cryptographic	This module includes functions that generates cryptographic random numbers.
RNG Non-cryptographic	This module includes functions that generates non cryptographic random numbers.
Radio Configuration	This module includes the platform abstraction for radio configuration.
Radio Extension	This module includes the Silicon Labs extension to the openthread platform radio interface.
Radio Operation	This module includes the platform abstraction for radio operations.
Radio Types	This module includes the platform abstraction for a radio frame.
Raw Link	This module includes functions that control the raw link-layer configuration.
Router/Leader	This module includes functions for Thread Routers and Leaders.
SNTP	This module includes functions that control SNTP communication.
SPI Slave	This module includes the platform abstraction for SPI slave communication.
SRP	This module includes functions that control SRP client behavior.
Server	This module includes functions to manage local network data with the OpenThread Server.
Settings	This module includes the platform abstraction for non-volatile storage of settings.
TCP	This module includes functions that control TCP communication.
TCP Abstractions	This module includes easy-to-use abstractions on top of the base TCP API.
TREL - Platform	This module includes the platform abstraction for Thread Radio Encapsulation Link (TREL) using DNS-SD and UDP/IPv6.
TREL - Thread Stack	This module defines Thread Radio Encapsulation Link (TREL) APIs for Thread Over Infrastructure.
Tasklets	This module includes functions that control the Thread stack's execution.
Time Service	This module includes the platform abstraction for the time service.
Toolchain	This module defines a toolchain abstraction layer through macros.
UDP	This module includes functions that control UDP communication.
UDP Forward	This module includes functions for UDP forward feature.

API Reference

API Reference

This module includes the application programming interface to the OpenThread stack.

Modules

Error

Execution

IPv6 Networking

Link

Message

Multi Radio Link

TREL - Thread Stack

Thread

Add-Ons

Error

Error

This module includes error definitions used in OpenThread.

Typedefs

typedef enum OT_MUST_USE_RESULT otError	otError Represents error codes used throughout OpenThread.

Functions

const char *	otThreadErrorToString(otError aError) Converts an otError enum into a string.

Typedef Documentation

otError

```
enum OT_MUST_USE_RESULT otError
```

Represents error codes used throughout OpenThread.

Definition at line 251 of file include/openthread/error.h

Function Documentation

otThreadErrorToString

```
const char * otThreadErrorToString (otError aError)
```

Converts an otError enum into a string.

Parameters

[in]	aError	An otError enum.

Returns

- A string representation of an otError.

Definition at line 261 of file include/openthread/error.h

Execution

Execution

Modules

Instance

Tasklets

Instance

Instance

This module includes functions that control the OpenThread Instance.

Typedefs

typedef struct otInstance
Represents the OpenThread instance structure.

typedef uint32_t otChangedFlags
Represents a bit-field indicating specific state/configuration that has changed.

typedef void(* otStateChangedCallback)(otChangedFlags aFlags, void *aContext)
Pointer is called to notify certain configuration or state changes within OpenThread.

Functions

otInstance * otInstanceInit(void *aInstanceBuffer, size_t *aInstanceBufferSize)
Initializes the OpenThread library.

otInstance * otInstanceInitSingle(void)
Initializes the static single instance of the OpenThread library.

uint32_t otInstanceGetId(otInstance *aInstance)
Gets the instance identifier.

bool otInstanceIsInitialized(otInstance *aInstance)
Indicates whether or not the instance is valid/initialized.

void otInstanceFinalize(otInstance *aInstance)
Disables the OpenThread library.

uint64_t otInstanceGetUptime(otInstance *aInstance)
Returns the current instance uptime (in msec).

void otInstanceGetUptimeAsString(otInstance *aInstance, char *aBuffer, uint16_t aSize)
Returns the current instance uptime as a human-readable string.

otError otSetStateChangedCallback(otInstance *aInstance, otStateChangedCallback aCallback, void *aContext)
Registers a callback to indicate when certain configuration or state changes within OpenThread.

void otRemoveStateChangeCallback(otInstance *aInstance, otStateChangedCallback aCallback, void *aContext)
Removes a callback to indicate when certain configuration or state changes within OpenThread.

void otInstanceReset(otInstance *aInstance)
Triggers a platform reset.

otError otInstanceResetToBootloader(otInstance *aInstance)
Triggers a platform reset to bootloader mode, if supported.

void otInstanceFactoryReset(otInstance *aInstance)
Deletes all the settings stored on non-volatile memory, and then triggers a platform reset.

void otInstanceResetRadioStack(otInstance *aInstance)
Resets the internal states of the OpenThread radio stack.

otError otInstanceErasePersistentInfo(otInstance *aInstance)
Erases all the OpenThread persistent info (network settings) stored on non-volatile memory.

const char * otGetVersionString(void)
Gets the OpenThread version string.

const char * otGetRadioVersionString(otInstance *aInstance)
Gets the OpenThread radio version string.

Macros

#define OT_UPTIME_STRING_SIZE 24
Recommended size for string representation of uptime.

#define OT_CHANGED_IP6_ADDRESS_ADDED (1U << 0)
IPv6 address was added.

#define OT_CHANGED_IP6_ADDRESS_REMOVED (1U << 1)
IPv6 address was removed.

#define OT_CHANGED_THREAD_ROLE (1U << 2)
Role (disabled, detached, child, router, leader) changed.

#define OT_CHANGED_THREAD_LL_ADDR (1U << 3)
The link-local address changed.

#define OT_CHANGED_THREAD_ML_ADDR (1U << 4)
The mesh-local address changed.

#define OT_CHANGED_THREAD_RLOC_ADDED (1U << 5)
RLOC was added.

#define OT_CHANGED_THREAD_RLOC_REMOVED (1U << 6)
RLOC was removed.

#define OT_CHANGED_THREAD_PARTITION_ID (1U << 7)
Partition ID changed.

#define OT_CHANGED_THREAD_KEY_SEQUENCE_COUNTER (1U << 8)
Thread Key Sequence changed.

#define OT_CHANGED_THREAD_NETDATA (1U << 9)
Thread Network Data changed.

#define OT_CHANGED_THREAD_CHILD_ADDED (1U << 10)
Child was added.

#define OT_CHANGED_THREAD_CHILD_REMOVED (1U << 11)
Child was removed.

#define OT_CHANGED_IP6_MULTICAST_SUBSCRIBED (1U << 12)
Subscribed to a IPv6 multicast address.

#define OT_CHANGED_IP6_MULTICAST_UNSUBSCRIBED (1U << 13)
Unsubscribed from a IPv6 multicast address.

#define OT_CHANGED_THREAD_CHANNEL (1U << 14)
Thread network channel changed.

#define OT_CHANGED_THREAD_PANID (1U << 15)
Thread network PAN Id changed.

#define OT_CHANGED_THREAD_NETWORK_NAME (1U << 16)
Thread network name changed.

#define OT_CHANGED_THREAD_EXT_PANID (1U << 17)
Thread network extended PAN ID changed.

#define OT_CHANGED_NETWORK_KEY (1U << 18)
Network key changed.

#define OT_CHANGED_PSKC (1U << 19)
PSKc changed.

#define OT_CHANGED_SECURITY_POLICY (1U << 20)
Security Policy changed.

#define OT_CHANGED_CHANNEL_MANAGER_NEW_CHANNEL (1U << 21)
Channel Manager new pending Thread channel changed.

#define OT_CHANGED_SUPPORTED_CHANNEL_MASK (1U << 22)
Supported channel mask changed.

#define OT_CHANGED_COMMISSIONER_STATE (1U << 23)
Commissioner state changed.

#define OT_CHANGED_THREAD_NETIF_STATE (1U << 24)
Thread network interface state changed.

#define OT_CHANGED_THREAD_BACKBONE_ROUTER_STATE (1U << 25)
Backbone Router state changed.

#define OT_CHANGED_THREAD_BACKBONE_ROUTER_LOCAL (1U << 26)
Local Backbone Router configuration changed.

#define OT_CHANGED_JOINER_STATE (1U << 27)
Joiner state changed.

#define OT_CHANGED_ACTIVE_DATASET (1U << 28)
Active Operational Dataset changed.

#define OT_CHANGED_PENDING_DATASET (1U << 29)
Pending Operational Dataset changed.

#define OT_CHANGED_NAT64_TRANSLATOR_STATE (1U << 30)
The state of NAT64 translator changed.

#define OT_CHANGED_PARENT_LINK_QUALITY (1U << 31)
Parent link quality changed.

Typedef Documentation

otInstance

```
typedef struct otInstance otInstance
```

Represents the OpenThread instance structure.

Definition at line 71 of file include/openthread/instance.h

otChangedFlags

```
typedef uint32_t otChangedFlags
```

Represents a bit-field indicating specific state/configuration that has changed.

See `OT_CHANGED_*` definitions.

Definition at line `212` of file `include/openthread/instance.h`

otStateChangedCallback

```
typedef void(* otStateChangedCallback) (otChangedFlags aFlags, void *aContext) )(otChangedFlags aFlags, void *aContext)
```

Pointer is called to notify certain configuration or state changes within OpenThread.

Parameters

[in]	aFlags	A bit-field indicating specific state that has changed. See `OT_CHANGED_*` definitions.
[in]	aContext	A pointer to application-specific context.

Definition at line `221` of file `include/openthread/instance.h`

Function Documentation

otInstanceInit

```
otInstance * otInstanceInit (void *aInstanceBuffer, size_t *aInstanceBufferSize)
```

Initializes the OpenThread library.

Parameters

[in]	aInstanceBuffer	The buffer for OpenThread to use for allocating the otInstance structure.
[inout]	aInstanceBufferSize	On input, the size of aInstanceBuffer. On output, if not enough space for otInstance, the number of bytes required for otInstance.

Initializes OpenThread and prepares it for subsequent OpenThread API calls. This function must be called before any other calls to OpenThread.

Is available and can only be used when support for multiple OpenThread instances is enabled.

Returns

- A pointer to the new OpenThread instance.

See Also

- otInstanceFinalize

Definition at line `90` of file `include/openthread/instance.h`

otInstanceInitSingle

```
otInstance * otInstanceInitSingle (void)
```

Initializes the static single instance of the OpenThread library.

Parameters

N/A		

Initializes OpenThread and prepares it for subsequent OpenThread API calls. This function must be called before any other calls to OpenThread.

Is available and can only be used when support for multiple OpenThread instances is disabled.

Returns

- A pointer to the single OpenThread instance.

Definition at line `103` of file `include/openthread/instance.h`

otInstanceGetId

```
uint32_t otInstanceGetId (otInstance *aInstance)
```

Gets the instance identifier.

Parameters

N/A	aInstance	

The instance identifier is set to a random value when the instance is constructed, and then its value will not change after initialization.

Returns

- The instance identifier.

Definition at line `114` of file `include/openthread/instance.h`

otInstanceIsInitialized

```
bool otInstanceIsInitialized (otInstance *aInstance)
```

Indicates whether or not the instance is valid/initialized.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The instance is considered valid if it is acquired and initialized using either `otInstanceInitSingle()` (in single instance case) or `otInstanceInit()` (in multi instance case). A subsequent call to `otInstanceFinalize()` causes the instance to be considered as uninitialized.

Returns

- TRUE if the given instance is valid/initialized, FALSE otherwise.

Definition at line `128` of file `include/openthread/instance.h`

otInstanceFinalize

```
void otInstanceFinalize (otInstance *aInstance)
```

Disables the OpenThread library.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Call this function when OpenThread is no longer in use.

Definition at line `138` of file `include/openthread/instance.h`

otInstanceGetUptime

```
uint64_t otInstanceGetUptime (otInstance *aInstance)
```

Returns the current instance uptime (in msec).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Requires `OPENTHREAD_CONFIG_UPTIME_ENABLE` to be enabled.

The uptime is given as number of milliseconds since OpenThread instance was initialized.

Returns

- The uptime (number of milliseconds).

Definition at line `152` of file `include/openthread/instance.h`

otInstanceGetUptimeAsString

```
void otInstanceGetUptimeAsString (otInstance *aInstance, char *aBuffer, uint16_t aSize)
```

Returns the current instance uptime as a human-readable string.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aBuffer	A pointer to a char array to output the string.
[in]	aSize	The size of `aBuffer` (in bytes). Recommended to use `OT_UPTIME_STRING_SIZE`.

Requires `OPENTHREAD_CONFIG_UPTIME_ENABLE` to be enabled.

The string follows the format "<hh>:<mm>:<ss>.<mmmm>" for hours, minutes, seconds and millisecond (if uptime is shorter than one day) or "<dd>d.<hh>:<mm>:<ss>.<mmmm>" (if longer than a day).

If the resulting string does not fit in `aBuffer` (within its `aSize` characters), the string will be truncated but the outputted string is always null-terminated.

Definition at line `172` of file `include/openthread/instance.h`

otSetStateChangedCallback

```
otError otSetStateChangedCallback (otInstance *aInstance, otStateChangedCallback aCallback, void *aContext)
```

Registers a callback to indicate when certain configuration or state changes within OpenThread.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function that is called with certain configuration or state changes.
[in]	aContext	A pointer to application-specific context.

Definition at line 235 of file include/openthread/instance.h

otRemoveStateChangeCallback

```
void otRemoveStateChangeCallback (otInstance *aInstance, otStateChangedCallback aCallback, void *aContext)
```

Removes a callback to indicate when certain configuration or state changes within OpenThread.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function that is called with certain configuration or state changes.
[in]	aContext	A pointer to application-specific context.

Definition at line 245 of file include/openthread/instance.h

otInstanceReset

```
void otInstanceReset (otInstance *aInstance)
```

Triggers a platform reset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The reset process ensures that all the OpenThread state/info (stored in volatile memory) is erased. Note that the otPlatformReset does not erase any persistent state/info saved in non-volatile memory.

Definition at line 256 of file include/openthread/instance.h

otInstanceResetToBootloader

```
otError otInstanceResetToBootloader (otInstance *aInstance)
```

Triggers a platform reset to bootloader mode, if supported.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Requires OPENTHREAD_CONFIG_PLATFORM_BOOTLOADER_MODE_ENABLE .

Definition at line 270 of file include/openthread/instance.h

otInstanceFactoryReset

```
void otInstanceFactoryReset (otInstance *aInstance)
```

Deletes all the settings stored on non-volatile memory, and then triggers a platform reset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `278` of file `include/openthread/instance.h`

otInstanceResetRadioStack

```
void otInstanceResetRadioStack (otInstance *aInstance)
```

Resets the internal states of the OpenThread radio stack.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Callbacks and configurations are preserved.

This API is only available under radio builds (`OPENTHREAD_RADIO = 1`).

Definition at line `290` of file `include/openthread/instance.h`

otInstanceErasePersistentInfo

```
otError otInstanceErasePersistentInfo (otInstance *aInstance)
```

Erases all the OpenThread persistent info (network settings) stored on non-volatile memory.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Erase is successful only if the device is in `disabled` state/role.

Definition at line `302` of file `include/openthread/instance.h`

otGetVersionString

```
const char * otGetVersionString (void)
```

Gets the OpenThread version string.

Parameters

N/A		

Returns

- A pointer to the OpenThread version.

Definition at line `310` of file `include/openthread/instance.h`

otGetRadioVersionString

```
const char * otGetRadioVersionString (otInstance *aInstance)
```

Gets the OpenThread radio version string.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the OpenThread radio version.

Definition at line 320 of file include/openthread/instance.h

Macro Definition Documentation

OT_UPTIME_STRING_SIZE

```
#define OT_UPTIME_STRING_SIZE
```

Value:

```
24
```

Recommended size for string representation of uptime.

Definition at line 154 of file include/openthread/instance.h

OT_CHANGED_IP6_ADDRESS_ADDED

```
#define OT_CHANGED_IP6_ADDRESS_ADDED
```

Value:

```
(1U << 0)
```

IPv6 address was added.

Definition at line 174 of file include/openthread/instance.h

OT_CHANGED_IP6_ADDRESS_REMOVED

```
#define OT_CHANGED_IP6_ADDRESS_REMOVED
```

Value:

```
(1U << 1)
```

IPv6 address was removed.

Definition at line 175 of file include/openthread/instance.h

OT_CHANGED_THREAD_ROLE

```
#define OT_CHANGED_THREAD_ROLE
```

Value:

```
(1U << 2)
```

Role (disabled, detached, child, router, leader) changed.

Definition at line 176 of file include/openthread/instance.h

OT_CHANGED_THREAD_LL_ADDR

```
#define OT_CHANGED_THREAD_LL_ADDR
```

Value:

```
(1U << 3)
```

The link-local address changed.

Definition at line 177 of file include/openthread/instance.h

OT_CHANGED_THREAD_ML_ADDR

```
#define OT_CHANGED_THREAD_ML_ADDR
```

Value:

```
(1U << 4)
```

The mesh-local address changed.

Definition at line 178 of file include/openthread/instance.h

OT_CHANGED_THREAD_RLOC_ADDED

```
#define OT_CHANGED_THREAD_RLOC_ADDED
```

Value:

```
(1U << 5)
```

RLOC was added.

Definition at line 179 of file include/openthread/instance.h

OT_CHANGED_THREAD_RLOC_REMOVED

```
#define OT_CHANGED_THREAD_RLOC_REMOVED
```

Value:

```
(1U << 6)
```

RLOC was removed.

Definition at line 180 of file include/openthread/instance.h

OT_CHANGED_THREAD_PARTITION_ID

```
#define OT_CHANGED_THREAD_PARTITION_ID
```

Value:

```
(1U << 7)
```

Partition ID changed.

Definition at line 181 of file include/openthread/instance.h

OT_CHANGED_THREAD_KEY_SEQUENCE_COUNTER

```
#define OT_CHANGED_THREAD_KEY_SEQUENCE_COUNTER
```

Value:

```
(1U << 8)
```

Thread Key Sequence changed.

Definition at line 182 of file include/openthread/instance.h

OT_CHANGED_THREAD_NETDATA

```
#define OT_CHANGED_THREAD_NETDATA
```

Value:

```
(1U << 9)
```

Thread Network Data changed.

Definition at line 183 of file include/openthread/instance.h

OT_CHANGED_THREAD_CHILD_ADDED

```
#define OT_CHANGED_THREAD_CHILD_ADDED
```

Value:

```
(1U << 10)
```

Child was added.

Definition at line 184 of file include/openthread/instance.h

OT_CHANGED_THREAD_CHILD_REMOVED

```
#define OT_CHANGED_THREAD_CHILD_REMOVED
```

Value:

```
(1U << 11)
```

Child was removed.

Definition at line 185 of file include/openthread/instance.h

OT_CHANGED_IP6_MULTICAST_SUBSCRIBED

```
#define OT_CHANGED_IP6_MULTICAST_SUBSCRIBED
```

Value:

```
(1U << 12)
```

Subscribed to a IPv6 multicast address.

Definition at line 186 of file include/openthread/instance.h

OT_CHANGED_IP6_MULTICAST_UNSUBSCRIBED

```
#define OT_CHANGED_IP6_MULTICAST_UNSUBSCRIBED
```

Value:

```
(1U << 13)
```

Unsubscribed from a IPv6 multicast address.

Definition at line 187 of file include/openthread/instance.h

OT_CHANGED_THREAD_CHANNEL

```
#define OT_CHANGED_THREAD_CHANNEL
```

Value:

```
(1U << 14)
```

Thread network channel changed.

Definition at line 188 of file include/openthread/instance.h

OT_CHANGED_THREAD_PANID

```
#define OT_CHANGED_THREAD_PANID
```

Value:

```
(1U << 15)
```

Thread network PAN Id changed.

Definition at line 189 of file include/openthread/instance.h

OT_CHANGED_THREAD_NETWORK_NAME

```
#define OT_CHANGED_THREAD_NETWORK_NAME
```

Value:

```
(1U << 16)
```

Thread network name changed.

Definition at line 190 of file include/openthread/instance.h

OT_CHANGED_THREAD_EXT_PANID

```
#define OT_CHANGED_THREAD_EXT_PANID
```

Value:

```
(1U << 17)
```

Thread network extended PAN ID changed.

Definition at line 191 of file include/openthread/instance.h

OT_CHANGED_NETWORK_KEY

```
#define OT_CHANGED_NETWORK_KEY
```

Value:

```
(1U << 18)
```

Network key changed.

Definition at line 192 of file include/openthread/instance.h

OT_CHANGED_PSKC

```
#define OT_CHANGED_PSKC
```

Value:

```
(1U << 19)
```

PSKc changed.

Definition at line 193 of file include/openthread/instance.h

OT_CHANGED_SECURITY_POLICY

```
#define OT_CHANGED_SECURITY_POLICY
```

Value:

```
(1U << 20)
```

Security Policy changed.

Definition at line 194 of file include/openthread/instance.h

OT_CHANGED_CHANNEL_MANAGER_NEW_CHANNEL

```
#define OT_CHANGED_CHANNEL_MANAGER_NEW_CHANNEL
```

Value:

```
(1U << 21)
```

Channel Manager new pending Thread channel changed.

Definition at line 195 of file include/openthread/instance.h

OT_CHANGED_SUPPORTED_CHANNEL_MASK

```
#define OT_CHANGED_SUPPORTED_CHANNEL_MASK
```

Value:

```
(1U << 22)
```

Supported channel mask changed.

Definition at line 196 of file include/openthread/instance.h

OT_CHANGED_COMMISSIONER_STATE

```
#define OT_CHANGED_COMMISSIONER_STATE
```

Value:

```
(1U << 23)
```

Commissioner state changed.

Definition at line 197 of file include/openthread/instance.h

OT_CHANGED_THREAD_NETIF_STATE

```
#define OT_CHANGED_THREAD_NETIF_STATE
```

Value:

```
(1U << 24)
```

Thread network interface state changed.

Definition at line 198 of file include/openthread/instance.h

OT_CHANGED_THREAD_BACKBONE_ROUTER_STATE

```
#define OT_CHANGED_THREAD_BACKBONE_ROUTER_STATE
```

Value:

```
(1U << 25)
```

Backbone Router state changed.

Definition at line 199 of file include/openthread/instance.h

OT_CHANGED_THREAD_BACKBONE_ROUTER_LOCAL

```
#define OT_CHANGED_THREAD_BACKBONE_ROUTER_LOCAL
```

Value:

```
(1U << 26)
```

Local Backbone Router configuration changed.

Definition at line 200 of file include/openthread/instance.h

OT_CHANGED_JOINER_STATE

```
#define OT_CHANGED_JOINER_STATE
```

Value:

```
(1U << 27)
```

Joiner state changed.

Definition at line 201 of file include/openthread/instance.h

OT_CHANGED_ACTIVE_DATASET

```
#define OT_CHANGED_ACTIVE_DATASET
```

Value:

```
(1U << 28)
```

Active Operational Dataset changed.

Definition at line 202 of file include/openthread/instance.h

OT_CHANGED_PENDING_DATASET

```
#define OT_CHANGED_PENDING_DATASET
```

Value:

```
(1U << 29)
```

Pending Operational Dataset changed.

Definition at line 203 of file include/openthread/instance.h

OT_CHANGED_NAT64_TRANSLATOR_STATE

```
#define OT_CHANGED_NAT64_TRANSLATOR_STATE
```

Value:

```
(1U << 30)
```

The state of NAT64 translator changed.

Definition at line 204 of file include/openthread/instance.h

OT_CHANGED_PARENT_LINK_QUALITY

```
#define OT_CHANGED_PARENT_LINK_QUALITY
```

Value:

```
(1U << 31)
```

Parent link quality changed.

Definition at line 205 of file include/openthread/instance.h

Tasklets

Tasklets

This module includes functions that control the Thread stack's execution.

Functions

void	otTaskletsProcess(otInstance *aInstance) Run all queued OpenThread tasklets at the time this is called.
bool	otTaskletsArePending(otInstance *aInstance) Indicates whether or not OpenThread has tasklets pending.
void	otTaskletsSignalPending(otInstance *aInstance) OpenThread calls this function when the tasklet queue transitions from empty to non-empty.

Function Documentation

otTaskletsProcess

```
void otTaskletsProcess (otInstance *aInstance)
```

Run all queued OpenThread tasklets at the time this is called.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `60` of file `include/openthread/tasklet.h`

otTaskletsArePending

```
bool otTaskletsArePending (otInstance *aInstance)
```

Indicates whether or not OpenThread has tasklets pending.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `71` of file `include/openthread/tasklet.h`

otTaskletsSignalPending

```
void otTaskletsSignalPending (otInstance *aInstance)
```

OpenThread calls this function when the tasklet queue transitions from empty to non-empty.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 79 of file include/openthread/tasklet.h

IPv6 Networking

IPv6 Networking

Modules

- DNS
- DNS-SD Server
- ICMPv6
- IPv6
- NAT64
- SRP
- Ping Sender
- TCP
- UDP

DNS

DNS

This module includes functions that control DNS communication.

The functions in this module are available only if feature `OPENTHREAD_CONFIG_DNS_CLIENT_ENABLE` is enabled.

Modules

otDnsTxtEntry

otDnsTxtEntryIterator

otDnsQueryConfig

otDnsServiceInfo

Enumerations

enum otDnsRecursionFlag {

OT_DNS_FLAG_UNSPECIFIED = 0
OT_DNS_FLAG_RECURSION_DESIRED = 1
OT_DNS_FLAG_NO_RECURSION = 2

}

Type represents the "Recursion Desired" (RD) flag in an `otDnsQueryConfig`.

enum otDnsNat64Mode {

OT_DNS_NAT64_UNSPECIFIED = 0
OT_DNS_NAT64_ALLOW = 1
OT_DNS_NAT64_DISALLOW = 2

}

Type represents the NAT64 mode in an `otDnsQueryConfig`.

enum otDnsServiceMode {

OT_DNS_SERVICE_MODE_UNSPECIFIED = 0
OT_DNS_SERVICE_MODE_SRV = 1
OT_DNS_SERVICE_MODE_TXT = 2
OT_DNS_SERVICE_MODE_SRV_TXT = 3
OT_DNS_SERVICE_MODE_SRV_TXT_SEPARATE = 4
OT_DNS_SERVICE_MODE_SRV_TXT_OPTIMIZE = 5

}

Type represents the service resolution mode in an `otDnsQueryConfig`.

enum otDnsTransportProto {

OT_DNS_TRANSPORT_UNSPECIFIED = 0
OT_DNS_TRANSPORT_UDP = 1
OT_DNS_TRANSPORT_TCP = 2

}

Type represents the DNS transport protocol in an `otDnsQueryConfig`.

Typedefs

typedef struct otDnsTxtEntry	otDnsTxtEntry Represents a TXT record entry representing a key/value pair (RFC 6763 - section 6.3).
typedef struct otDnsTxtEntryIterator	otDnsTxtEntryIterator Represents an iterator for TXT record entries (key/value pairs).
typedef struct otDnsQueryConfig	otDnsQueryConfig Represents a DNS query configuration.
typedef struct otDnsAddressResponse	otDnsAddressResponse An opaque representation of a response to an address resolution DNS query.
typedef void(*	otDnsAddressCallback)(otError aError, const otDnsAddressResponse *aResponse, void *aContext) Pointer is called when a DNS response is received for an address resolution query.
typedef struct otDnsBrowseResponse	otDnsBrowseResponse An opaque representation of a response to a browse (service instance enumeration) DNS query.
typedef void(*	otDnsBrowseCallback)(otError aError, const otDnsBrowseResponse *aResponse, void *aContext) Pointer is called when a DNS response is received for a browse (service instance enumeration) query.
typedef struct otDnsServiceInfo	otDnsServiceInfo Provides info for a DNS service instance.
typedef struct otDnsServiceResponse	otDnsServiceResponse An opaque representation of a response to a service instance resolution DNS query.
typedef void(*	otDnsServiceCallback)(otError aError, const otDnsServiceResponse *aResponse, void *aContext) Pointer is called when a DNS response is received for a service instance resolution query.

Functions

void	otDnsInitTxtEntryIterator(otDnsTxtEntryIterator *aIterator, const uint8_t *aTxtData, uint16_t aTxtDataLength) Initializes a TXT record iterator.
otError	otDnsGetNextTxtEntry(otDnsTxtEntryIterator *aIterator, otDnsTxtEntry *aEntry) Parses the TXT data from an iterator and gets the next TXT record entry (key/value pair).
otError	otDnsEncodeTxtData(const otDnsTxtEntry *aTxtEntries, uint16_t aNumTxtEntries, uint8_t *aTxtData, uint16_t *aTxtDataLength) Encodes a given list of TXT record entries (key/value pairs) into TXT data (following format specified by RFC 6763).
void	otDnsSetNameCompressionEnabled(bool aEnabled) Enables/disables the "DNS name compression" mode.
bool	otDnsIsNameCompressionEnabled(void) Indicates whether the "DNS name compression" mode is enabled or not.
const otDnsQueryConfig *	otDnsClientGetDefaultConfig(otInstance *aInstance) Gets the current default query config used by DNS client.
void	otDnsClientSetDefaultConfig(otInstance *aInstance, const otDnsQueryConfig *aConfig) Sets the default query config on DNS client.

otError otDnsClientResolveAddress(otInstance *aInstance, const char *aHostName, otDnsAddressCallback aCallback, void *aContext, const otDnsQueryConfig *aConfig)
Sends an address resolution DNS query for AAAA (IPv6) record(s) for a given host name.

otError otDnsClientResolveIp4Address(otInstance *aInstance, const char *aHostName, otDnsAddressCallback aCallback, void *aContext, const otDnsQueryConfig *aConfig)
Sends an address resolution DNS query for A (IPv4) record(s) for a given host name.

otError otDnsAddressResponseGetHostName(const otDnsAddressResponse *aResponse, char *aNameBuffer, uint16_t aNameBufferSize)
Gets the full host name associated with an address resolution DNS response.

otError otDnsAddressResponseGetAddress(const otDnsAddressResponse *aResponse, uint16_t aIndex, otIp6Address *aAddress, uint32_t *aTtl)
Gets an IPv6 address associated with an address resolution DNS response.

otError otDnsClientBrowse(otInstance *aInstance, const char *aServiceName, otDnsBrowseCallback aCallback, void *aContext, const otDnsQueryConfig *aConfig)
Sends a DNS browse (service instance enumeration) query for a given service name.

otError otDnsBrowseResponseGetServiceName(const otDnsBrowseResponse *aResponse, char *aNameBuffer, uint16_t aNameBufferSize)
Gets the service name associated with a DNS browse (service instance enumeration) response.

otError otDnsBrowseResponseGetServiceInstance(const otDnsBrowseResponse *aResponse, uint16_t aIndex, char *aLabelBuffer, uint8_t aLabelBufferSize)
Gets a service instance associated with a DNS browse (service instance enumeration) response.

otError otDnsBrowseResponseGetServiceInfo(const otDnsBrowseResponse *aResponse, const char *aInstanceLabel, otDnsServiceInfo *aServiceInfo)
Gets info for a service instance from a DNS browse (service instance enumeration) response.

otError otDnsBrowseResponseGetHostAddress(const otDnsBrowseResponse *aResponse, const char *aHostName, uint16_t aIndex, otIp6Address *aAddress, uint32_t *aTtl)
Gets the host IPv6 address from a DNS browse (service instance enumeration) response.

otError otDnsClientResolveService(otInstance *aInstance, const char *aInstanceLabel, const char *aServiceName, otDnsServiceCallback aCallback, void *aContext, const otDnsQueryConfig *aConfig)
Starts a DNS service instance resolution for a given service instance.

otError otDnsClientResolveServiceAndHostAddress(otInstance *aInstance, const char *aInstanceLabel, const char *aServiceName, otDnsServiceCallback aCallback, void *aContext, const otDnsQueryConfig *aConfig)
Starts a DNS service instance resolution for a given service instance, with a potential follow-up address resolution for the host name discovered for the service instance.

otError otDnsServiceResponseGetServiceName(const otDnsServiceResponse *aResponse, char *aLabelBuffer, uint8_t aLabelBufferSize, char *aNameBuffer, uint16_t aNameBufferSize)
Gets the service instance name associated with a DNS service instance resolution response.

otError otDnsServiceResponseGetServiceInfo(const otDnsServiceResponse *aResponse, otDnsServiceInfo *aServiceInfo)
Gets info for a service instance from a DNS service instance resolution response.

otError otDnsServiceResponseGetHostAddress(const otDnsServiceResponse *aResponse, const char *aHostName, uint16_t aIndex, otIp6Address *aAddress, uint32_t *aTtl)
Gets the host IPv6 address from a DNS service instance resolution response.

Macros

#define OT_DNS_MAX_NAME_SIZE 255
Maximum name string size (includes null char at the end of string).

#define OT_DNS_MAX_LABEL_SIZE 64
Maximum label string size (include null char at the end of string).

#define OT_DNS_TXT_KEY_MIN_LENGTH 1
Minimum length of TXT record key string (RFC 6763 - section 6.4).

#define OT_DNS_TXT_KEY_MAX_LENGTH 9
Recommended maximum length of TXT record key string (RFC 6763 - section 6.4).

Enumeration Documentation

otDnsRecursionFlag

> otDnsRecursionFlag

Type represents the "Recursion Desired" (RD) flag in an `otDnsQueryConfig` .

Enumerator	
OT_DNS_FLAG_UNSPECIFIED	Indicates the flag is not specified.
OT_DNS_FLAG_RECURSION_DESIRED	Indicates DNS name server can resolve the query recursively.
OT_DNS_FLAG_NO_RECURSION	Indicates DNS name server can not resolve the query recursively.

Definition at line `62` of file `include/openthread/dns_client.h`

otDnsNat64Mode

> otDnsNat64Mode

Type represents the NAT64 mode in an `otDnsQueryConfig` .

The NAT64 mode indicates whether to allow or disallow NAT64 address translation during DNS client address resolution. This mode is only used when `OPENTHREAD_CONFIG_DNS_CLIENT_NAT64_ENABLE` is enabled.

Enumerator	
OT_DNS_NAT64_UNSPECIFIED	NAT64 mode is not specified. Use default NAT64 mode.
OT_DNS_NAT64_ALLOW	Allow NAT64 address translation during DNS client address resolution.
OT_DNS_NAT64_DISALLOW	Do not allow NAT64 address translation during DNS client address resolution.

Definition at line `76` of file `include/openthread/dns_client.h`

otDnsServiceMode

> otDnsServiceMode

Type represents the service resolution mode in an `otDnsQueryConfig` .

This is only used during DNS client service resolution `otDnsClientResolveService()` . It determines which record types to query.

Enumerator	
OT_DNS_SERVICE_MODE_UNSPECIFIED	Mode is not specified. Use default service mode.
OT_DNS_SERVICE_MODE_SRV	Query for SRV record only.
OT_DNS_SERVICE_MODE_TXT	Query for TXT record only.

OT_DNS_SERVICE_MODE_SRV_TXT	Query for both SRV and TXT records in same message.
OT_DNS_SERVICE_MODE_SRV_TXT_SEPARATE	Query in parallel for SRV and TXT using separate messages.
OT_DNS_SERVICE_MODE_SRV_TXT_OPTIMIZE	Query for TXT/SRV together first, if fails then query separately.

Definition at line `90` of file `include/openthread/dns_client.h`

otDnsTransportProto

> otDnsTransportProto

Type represents the DNS transport protocol in an `otDnsQueryConfig`.

This `OT_DNS_TRANSPORT_TCP` is only supported when `OPENTHREAD_CONFIG_DNS_CLIENT_OVER_TCP_ENABLE` is enabled.

Enumerator	
OT_DNS_TRANSPORT_UNSPECIFIED	
OT_DNS_TRANSPORT_UDP	DNS transport is unspecified.
OT_DNS_TRANSPORT_TCP	DNS query should be sent via UDP.

Definition at line `106` of file `include/openthread/dns_client.h`

Typedef Documentation

otDnsTxtEntry

> typedef struct otDnsTxtEntry otDnsTxtEntry

Represents a TXT record entry representing a key/value pair (RFC 6763 - section 6.3).

The string buffers pointed to by `mKey` and `mValue` MUST persist and remain unchanged after an instance of such structure is passed to OpenThread (as part of `otSrpClientService` instance).

An array of `otDnsTxtEntry` entries are used in `otSrpClientService` to specify the full TXT record (a list of entries).

Definition at line `95` of file `include/openthread/dns.h`

otDnsTxtEntryIterator

> typedef struct otDnsTxtEntryIterator otDnsTxtEntryIterator

Represents an iterator for TXT record entries (key/value pairs).

The data fields in this structure are intended for use by OpenThread core and caller should not read or change them.

Definition at line `108` of file `include/openthread/dns.h`

otDnsQueryConfig

> typedef struct otDnsQueryConfig otDnsQueryConfig

Represents a DNS query configuration.

Any of the fields in this structure can be set to zero to indicate that it is not specified. How the unspecified fields are treated is determined by the function which uses the instance of `otDnsQueryConfig`.

Definition at line `129` of file `include/openthread/dns_client.h`

otDnsAddressResponse

```
typedef struct otDnsAddressResponse otDnsAddressResponse
```

An opaque representation of a response to an address resolution DNS query.

Pointers to instance of this type are provided from callback `otDnsAddressCallback`.

Definition at line `177` of file `include/openthread/dns_client.h`

otDnsAddressCallback

```
typedef void(* otDnsAddressCallback) (otError aError, const otDnsAddressResponse *aResponse, void *aContext) )
(otError aError, const otDnsAddressResponse *aResponse, void *aContext)
```

Pointer is called when a DNS response is received for an address resolution query.

Parameters

[in]	aError	The result of the DNS transaction.
[in]	aResponse	A pointer to the response (it is always non-NULL).
[in]	aContext	A pointer to application-specific context.

Within this callback the user can use `otDnsAddressResponseGet{Item}()` functions along with the `aResponse` pointer to get more info about the response.

The `aResponse` pointer can only be used within this callback and after returning from this function it will not stay valid, so the user MUST NOT retain the `aResponse` pointer for later use.

The `aError` can have the following:

- OT_ERROR_NONE A response was received successfully.
- OT_ERROR_ABORT A DNS transaction was aborted by stack.
- OT_ERROR_RESPONSE_TIMEOUT No DNS response has been received within timeout.

If the server rejects the address resolution request the error code from server is mapped as follow:

- (0) NOERROR Success (no error condition) -> OT_ERROR_NONE
- (1) FORMERR Server unable to interpret due to format error -> OT_ERROR_PARSE
- (2) SERVFAIL Server encountered an internal failure -> OT_ERROR_FAILED
- (3) NXDOMAIN Name that ought to exist, does not exist -> OT_ERROR_NOT_FOUND
- (4) NOTIMP Server does not support the query type (OpCode) -> OT_ERROR_NOT_IMPLEMENTED
- (5) REFUSED Server refused for policy/security reasons -> OT_ERROR_SECURITY
- (6) YXDOMAIN Some name that ought not to exist, does exist -> OT_ERROR_DUPLICATED
- (7) YXRRSET Some RRset that ought not to exist, does exist -> OT_ERROR_DUPLICATED
- (8) NXRRSET Some RRset that ought to exist, does not exist -> OT_ERROR_NOT_FOUND
- (9) NOTAUTH Service is not authoritative for zone -> OT_ERROR_SECURITY
- (10) NOTZONE A name is not in the zone -> OT_ERROR_PARSE
- (20) BADNAME Bad name -> OT_ERROR_PARSE
- (21) BADALG Bad algorithm -> OT_ERROR_SECURITY
- (22) BADTRUN Bad truncation -> OT_ERROR_PARSE
- Other response codes -> OT_ERROR_FAILED

Definition at line 217 of file include/openthread/dns_client.h

otDnsBrowseResponse

```
typedef struct otDnsBrowseResponse otDnsBrowseResponse
```

An opaque representation of a response to a browse (service instance enumeration) DNS query.

Pointers to instance of this type are provided from callback otDnsBrowseCallback .

Definition at line 323 of file include/openthread/dns_client.h

otDnsBrowseCallback

```
typedef void(* otDnsBrowseCallback) (otError aError, const otDnsBrowseResponse *aResponse, void *aContext) )
(otError aError, const otDnsBrowseResponse *aResponse, void *aContext)
```

Pointer is called when a DNS response is received for a browse (service instance enumeration) query.

Parameters

[in]	aError	The result of the DNS transaction.
[in]	aResponse	A pointer to the response (it is always non-NULL).
[in]	aContext	A pointer to application-specific context.

Within this callback the user can use otDnsBrowseResponseGet{Item}() functions along with the aResponse pointer to get more info about the response.

The aResponse pointer can only be used within this callback and after returning from this function it will not stay valid, so the user MUST NOT retain the aResponse pointer for later use.

For the full list of possible values for aError , please see otDnsAddressCallback() .

Definition at line 341 of file include/openthread/dns_client.h

otDnsServiceInfo

```
typedef struct otDnsServiceInfo otDnsServiceInfo
```

Provides info for a DNS service instance.

Definition at line 361 of file include/openthread/dns_client.h

otDnsServiceResponse

```
typedef struct otDnsServiceResponse otDnsServiceResponse
```

An opaque representation of a response to a service instance resolution DNS query.

Pointers to instance of this type are provided from callback otDnsAddressCallback .

Definition at line 497 of file include/openthread/dns_client.h

otDnsServiceCallback

typedef void(* otDnsServiceCallback) (otError aError, const otDnsServiceResponse *aResponse, void *aContext))
(otError aError, const otDnsServiceResponse *aResponse, void *aContext)

Pointer is called when a DNS response is received for a service instance resolution query.

Parameters

[in]	aError	The result of the DNS transaction.
[in]	aResponse	A pointer to the response (it is always non-NULL).
[in]	aContext	A pointer to application-specific context.

Within this callback the user can use `otDnsServiceResponseGet{Item}()` functions along with the `aResponse` pointer to get more info about the response.

The `aResponse` pointer can only be used within this callback and after returning from this function it will not stay valid, so the user MUST NOT retain the `aResponse` pointer for later use.

For the full list of possible values for `aError`, please see `otDnsAddressCallback()`.

Definition at line `515` of file `include/openthread/dns_client.h`

Function Documentation

otDnsInitTxtEntryIterator

void otDnsInitTxtEntryIterator (otDnsTxtEntryIterator *aIterator, const uint8_t *aTxtData, uint16_t aTxtDataLength)

Initializes a TXT record iterator.

Parameters

[in]	aIterator	A pointer to the iterator to initialize (MUST NOT be NULL).
[in]	aTxtData	A pointer to buffer containing the encoded TXT data.
[in]	aTxtDataLength	The length (number of bytes) of `aTxtData`.

The buffer pointer `aTxtData` and its content MUST persist and remain unchanged while `aIterator` object is being used.

Definition at line `121` of file `include/openthread/dns.h`

otDnsGetNextTxtEntry

otError otDnsGetNextTxtEntry (otDnsTxtEntryIterator *aIterator, otDnsTxtEntry *aEntry)

Parses the TXT data from an iterator and gets the next TXT record entry (key/value pair).

Parameters

[in]	aIterator	A pointer to the iterator (MUST NOT be NULL).
[out]	aEntry	A pointer to a `otDnsTxtEntry` structure to output the parsed/read entry (MUST NOT be NULL).

The `aIterator` MUST be initialized using `otDnsInitTxtEntryIterator()` before calling this function and the TXT data buffer used to initialize the iterator MUST persist and remain unchanged. Otherwise the behavior of this function is undefined.

If the parsed key string length is smaller than or equal to `OT_DNS_TXT_KEY_MAX_LENGTH` (recommended max key length) the key string is returned in `mKey` in `aEntry`. But if the key is longer, then `mKey` is set to NULL and the entire encoded TXT

entry string is returned in `mValue` and `mValueLength`.

Definition at line `142` of file `include/openthread/dns.h`

otDnsEncodeTxtData

```
otError otDnsEncodeTxtData (const otDnsTxtEntry *aTxtEntries, uint16_t aNumTxtEntries, uint8_t *aTxtData, uint16_t *aTxtDataLength)
```

Encodes a given list of TXT record entries (key/value pairs) into TXT data (following format specified by RFC 6763).

Parameters

[in]	aTxtEntries	Pointer to an array of `otDnsTxtEntry`.
[in]	aNumTxtEntries	Number of entries in `aTxtEntries` array.
[out]	aTxtData	A pointer to a buffer to output the encoded TXT data.
[inout]	aTxtDataLength	On input, size of buffer `aTxtData`. On output, length of the encoded TXT data.

Definition at line `157` of file `include/openthread/dns.h`

otDnsSetNameCompressionEnabled

```
void otDnsSetNameCompressionEnabled (bool aEnabled)
```

Enables/disables the "DNS name compression" mode.

Parameters

[in]	aEnabled	TRUE to enable the "DNS name compression" mode, FALSE to disable.

By default DNS name compression is enabled. When disabled, DNS names are appended as full and never compressed. This is applicable to OpenThread's DNS and SRP client/server modules.

This is intended for testing only and available when `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE` config is enabled.

Note that in the case `OPENTHREAD_CONFIG_MULTIPLE_INSTANCE_ENABLE` is used, this mode applies to all OpenThread instances (i.e., calling this function enables/disables the compression mode on all OpenThread instances).

Definition at line `176` of file `include/openthread/dns.h`

otDnsIsNameCompressionEnabled

```
bool otDnsIsNameCompressionEnabled (void)
```

Indicates whether the "DNS name compression" mode is enabled or not.

Parameters

N/A		

This is intended for testing only and available when `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE` config is enabled.

Returns

- TRUE if the "DNS name compression" mode is enabled, FALSE otherwise.

Definition at line 186 of file include/openthread/dns.h

otDnsClientGetDefaultConfig

```
const otDnsQueryConfig * otDnsClientGetDefaultConfig (otInstance *aInstance)
```

Gets the current default query config used by DNS client.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

When OpenThread stack starts, the default DNS query config is determined from a set of OT config options such as OPENTHREAD_CONFIG_DNS_CLIENT_DEFAULT_SERVER_IP6_ADDRESS , _DEFAULT_SERVER_PORT , _DEFAULT_RESPONSE_TIMEOUT , etc. (see config/dns_client.h for all related config options).

Returns

- A pointer to the current default config being used by DNS client.

Definition at line 143 of file include/openthread/dns_client.h

otDnsClientSetDefaultConfig

```
void otDnsClientSetDefaultConfig (otInstance *aInstance, const otDnsQueryConfig *aConfig)
```

Sets the default query config on DNS client.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aConfig	A pointer to the new query config to use as default.

Note

- Any ongoing query will continue to use the config from when it was started. The new default config will be used for any future DNS queries.

The aConfig can be NULL. In this case the default config will be set to the defaults from OT config options OPENTHREAD_CONFIG_DNS_CLIENT_DEFAULT_{} . This resets the default query config back to to the config when the OpenThread stack starts.

In a non-NULL aConfig , caller can choose to leave some of the fields in otDnsQueryConfig instance unspecified (value zero). The unspecified fields are replaced by the corresponding OT config option definitions OPENTHREAD_CONFIG_DNS_CLIENT_DEFAULT_{} to form the default query config.

When OPENTHREAD_CONFIG_DNS_CLIENT_DEFAULT_SERVER_ADDRESS_AUTO_SET_ENABLE is enabled, the server's IPv6 address in the default config is automatically set and updated by DNS client. This is done only when user does not explicitly set or specify it. This behavior requires SRP client and its auto-start feature to be enabled. SRP client will then monitor the Thread Network Data for DNS/SRP Service entries to select an SRP server. The selected SRP server address is also set as the DNS server address in the default config.

Definition at line 169 of file include/openthread/dns_client.h

otDnsClientResolveAddress

```
otError otDnsClientResolveAddress (otInstance *aInstance, const char *aHostName, otDnsAddressCallback aCallback,
void *aContext, const otDnsQueryConfig *aConfig)
```

Sends an address resolution DNS query for AAAA (IPv6) record(s) for a given host name.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aHostName	The host name for which to query the address (MUST NOT be NULL).
[in]	aCallback	A function pointer that shall be called on response reception or time-out.
[in]	aContext	A pointer to arbitrary context information.
[in]	aConfig	A pointer to the config to use for this query.

The `aConfig` can be NULL. In this case the default config (from `otDnsClientGetDefaultConfig()`) will be used as the config for this query. In a non-NULL `aConfig`, some of the fields can be left unspecified (value zero). The unspecified fields are then replaced by the values from the default config.

Definition at line `238` of file `include/openthread/dns_client.h`

otDnsClientResolveIp4Address

```
otError otDnsClientResolveIp4Address (otInstance *aInstance, const char *aHostName, otDnsAddressCallback aCallback,
void *aContext, const otDnsQueryConfig *aConfig)
```

Sends an address resolution DNS query for A (IPv4) record(s) for a given host name.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aHostName	The host name for which to query the address (MUST NOT be NULL).
[in]	aCallback	A function pointer that shall be called on response reception or time-out.
[in]	aContext	A pointer to arbitrary context information.
[in]	aConfig	A pointer to the config to use for this query.

Requires and is available when `OPENTHREAD_CONFIG_DNS_CLIENT_NAT64_ENABLE` is enabled.

When a successful response is received, the addresses are returned from `aCallback` as NAT64 IPv6 translated versions of the IPv4 addresses from the query response.

The `aConfig` can be NULL. In this case the default config (from `otDnsClientGetDefaultConfig()`) will be used as the config for this query. In a non-NULL `aConfig`, some of the fields can be left unspecified (value zero). The unspecified fields are then replaced by the values from the default config.

Definition at line `268` of file `include/openthread/dns_client.h`

otDnsAddressResponseGetHostName

```
otError otDnsAddressResponseGetHostName (const otDnsAddressResponse *aResponse, char *aNameBuffer, uint16_t
aNameBufferSize)
```

Gets the full host name associated with an address resolution DNS response.

Parameters

[in]	aResponse	A pointer to the response.
[out]	aNameBuffer	A buffer to char array to output the full host name (MUST NOT be NULL).
[in]	aNameBufferSize	The size of `aNameBuffer`.

MUST only be used from `otDnsAddressCallback`.

Definition at line `287` of file `include/openthread/dns_client.h`

otDnsAddressResponseGetAddress

> otError otDnsAddressResponseGetAddress (const otDnsAddressResponse *aResponse, uint16_t aIndex, otIp6Address *aAddress, uint32_t *aTtl)

Gets an IPv6 address associated with an address resolution DNS response.

Parameters

[in]	aResponse	A pointer to the response.
[in]	aIndex	The address record index to retrieve.
[out]	aAddress	A pointer to a IPv6 address to output the address (MUST NOT be NULL).
[out]	aTtl	A pointer to an `uint32_t` to output TTL for the address. It can be NULL if caller does not want to get the TTL.

MUST only be used from `otDnsAddressCallback`.

The response may include multiple IPv6 address records. `aIndex` can be used to iterate through the list of addresses. Index zero gets the first address and so on. When we reach end of the list, `OT_ERROR_NOT_FOUND` is returned.

Definition at line `312` of file `include/openthread/dns_client.h`

otDnsClientBrowse

> otError otDnsClientBrowse (otInstance *aInstance, const char *aServiceName, otDnsBrowseCallback aCallback, void *aContext, const otDnsQueryConfig *aConfig)

Sends a DNS browse (service instance enumeration) query for a given service name.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aServiceName	The service name to query for (MUST NOT be NULL).
[in]	aCallback	A function pointer that shall be called on response reception or time-out.
[in]	aContext	A pointer to arbitrary context information.
[in]	aConfig	A pointer to the config to use for this query.

Is available when `OPENTHREAD_CONFIG_DNS_CLIENT_SERVICE_DISCOVERY_ENABLE` is enabled.

The `aConfig` can be NULL. In this case the default config (from `otDnsClientGetDefaultConfig()`) will be used as the config for this query. In a non-NULL `aConfig`, some of the fields can be left unspecified (value zero). The unspecified fields are then replaced by the values from the default config.

Definition at line `382` of file `include/openthread/dns_client.h`

otDnsBrowseResponseGetServiceName

> otError otDnsBrowseResponseGetServiceName (const otDnsBrowseResponse *aResponse, char *aNameBuffer, uint16_t aNameBufferSize)

Gets the service name associated with a DNS browse (service instance enumeration) response.

Parameters

[in]	aResponse	A pointer to the response.
[out]	aNameBuffer	A buffer to char array to output the service name (MUST NOT be NULL).
[in]	aNameBufferSize	The size of `aNameBuffer`.

MUST only be used from `otDnsBrowseCallback`.

Definition at line `401` of file `include/openthread/dns_client.h`

otDnsBrowseResponseGetServiceInstance

> otError otDnsBrowseResponseGetServiceInstance (const otDnsBrowseResponse *aResponse, uint16_t aIndex, char *aLabelBuffer, uint8_t aLabelBufferSize)

Gets a service instance associated with a DNS browse (service instance enumeration) response.

Parameters

[in]	aResponse	A pointer to the response.
[in]	aIndex	The service instance record index to retrieve.
[out]	aLabelBuffer	A buffer to char array to output the service instance label (MUST NOT be NULL).
[in]	aLabelBufferSize	The size of `aLabelBuffer`.

MUST only be used from `otDnsBrowseCallback`.

The response may include multiple service instance records. `aIndex` can be used to iterate through the list. Index zero gives the the first record. When we reach end of the list, `OT_ERROR_NOT_FOUND` is returned.

Note that this function gets the service instance label and not the full service instance name which is of the form `<Instance>.<Service>.<Domain>`.

Definition at line `427` of file `include/openthread/dns_client.h`

otDnsBrowseResponseGetServiceInfo

> otError otDnsBrowseResponseGetServiceInfo (const otDnsBrowseResponse *aResponse, const char *aInstanceLabel, otDnsServiceInfo *aServiceInfo)

Gets info for a service instance from a DNS browse (service instance enumeration) response.

Parameters

[in]	aResponse	A pointer to the response.
[in]	aInstanceLabel	The service instance label (MUST NOT be NULL).
[out]	aServiceInfo	A `ServiceInfo` to output the service instance information (MUST NOT be NULL).

MUST only be used from `otDnsBrowseCallback`.

A browse DNS response can include SRV, TXT, and AAAA records for the service instances that are enumerated. This is a SHOULD and not a MUST requirement, and servers/resolvers are not required to provide this. This function attempts to retrieve this info for a given service instance when available.

If no matching SRV record is found in `aResponse`, `OT_ERROR_NOT_FOUND` is returned. In this case, no additional records (no TXT and/or AAAA) are read.

- If a matching SRV record is found in `aResponse`, `aServiceInfo` is updated and `OT_ERROR_NONE` is returned.
- If no matching TXT record is found in `aResponse`, `mTxtDataSize` in `aServiceInfo` is set to zero.
- If TXT data length is greater than `mTxtDataSize`, it is read partially and `mTxtDataTruncated` is set to true.
- If no matching AAAA record is found in `aResponse`, `mHostAddress is set to all zero or unspecified address.`
- `If there are multiple AAAA records for the host name in @p aResponse, mHostAddress is set to the first one. The other addresses can be retrieved using otDnsBrowseResponseGetHostAddress() `.

Definition at line `460` of file `include/openthread/dns_client.h`

otDnsBrowseResponseGetHostAddress

> otError otDnsBrowseResponseGetHostAddress (const otDnsBrowseResponse *aResponse, const char *aHostName, uint16_t aIndex, otIp6Address *aAddress, uint32_t *aTtl)

Gets the host IPv6 address from a DNS browse (service instance enumeration) response.

Parameters

[in]	aResponse	A pointer to the response.
[in]	aHostName	The host name to get the address (MUST NOT be NULL).
[in]	aIndex	The address record index to retrieve.
[out]	aAddress	A pointer to a IPv6 address to output the address (MUST NOT be NULL).
[out]	aTtl	A pointer to an `uint32_t` to output TTL for the address. It can be NULL if caller does not want to get the TTL.

MUST only be used from `otDnsBrowseCallback`.

The response can include zero or more IPv6 address records. `aIndex` can be used to iterate through the list of addresses. Index zero gets the first address and so on. When we reach end of the list, `OT_ERROR_NOT_FOUND` is returned.

Definition at line `485` of file `include/openthread/dns_client.h`

otDnsClientResolveService

> otError otDnsClientResolveService (otInstance *aInstance, const char *aInstanceLabel, const char *aServiceName, otDnsServiceCallback aCallback, void *aContext, const otDnsQueryConfig *aConfig)

Starts a DNS service instance resolution for a given service instance.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aInstanceLabel	The service instance label.
[in]	aServiceName	The service name (together with `aInstanceLabel` form full instance name).
[in]	aCallback	A function pointer that shall be called on response reception or time-out.
[in]	aContext	A pointer to arbitrary context information.
[in]	aConfig	A pointer to the config to use for this query.

Is available when `OPENTHREAD_CONFIG_DNS_CLIENT_SERVICE_DISCOVERY_ENABLE` is enabled.

The `aConfig` can be NULL. In this case the default config (from `otDnsClientGetDefaultConfig()`) will be used as the config for this query. In a non-NULL `aConfig`, some of the fields can be left unspecified (value zero). The unspecified fields are then

replaced by the values from the default config.

The function sends queries for SRV and/or TXT records for the given service instance. The `mServiceMode` field in `otDnsQueryConfig` determines which records to query (SRV only, TXT only, or both SRV and TXT) and how to perform the query (together in the same message, separately in parallel, or in optimized mode where client will try in the same message first and then separately if it fails to get a response).

The SRV record provides information about service port, priority, and weight along with the host name associated with the service instance. This function DOES NOT perform address resolution for the host name discovered from SRV record. The server/resolver may provide AAAA/A record(s) for the host name in the Additional Data section of the response to SRV/TXT query and this information can be retrieved using `otDnsServiceResponseGetServiceInfo()` in `otDnsServiceCallback`. Users of this API MUST NOT assume that host address will always be available from `otDnsServiceResponseGetServiceInfo()`.

Definition at line `550` of file `include/openthread/dns_client.h`

otDnsClientResolveServiceAndHostAddress

> otError otDnsClientResolveServiceAndHostAddress (otInstance *aInstance, const char *aInstanceLabel, const char *aServiceName, otDnsServiceCallback aCallback, void *aContext, const otDnsQueryConfig *aConfig)

Starts a DNS service instance resolution for a given service instance, with a potential follow-up address resolution for the host name discovered for the service instance.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aInstanceLabel	The service instance label.
[in]	aServiceName	The service name (together with `aInstanceLabel` form full instance name).
[in]	aCallback	A function pointer that shall be called on response reception or time-out.
[in]	aContext	A pointer to arbitrary context information.
[in]	aConfig	A pointer to the config to use for this query.

Is available when `OPENTHREAD_CONFIG_DNS_CLIENT_SERVICE_DISCOVERY_ENABLE` is enabled.

The `aConfig` can be NULL. In this case the default config (from `otDnsClientGetDefaultConfig()`) will be used as the config for this query. In a non-NULL `aConfig`, some of the fields can be left unspecified (value zero). The unspecified fields are then replaced by the values from the default config. This function cannot be used with `mServiceMode` in DNS config set to `OT_DNS_SERVICE_MODE_TXT` (i.e., querying for TXT record only) and will return `OT_ERROR_INVALID_ARGS`.

Behaves similarly to `otDnsClientResolveService()` sending queries for SRV and TXT records. However, if the server/resolver does not provide AAAA/A records for the host name in the response to SRV query (in the Additional Data section), it will perform host name resolution (sending an AAAA query) for the discovered host name from the SRV record. The callback `aCallback` is invoked when responses for all queries are received (i.e., both service and host address resolutions are finished).

Definition at line `587` of file `include/openthread/dns_client.h`

otDnsServiceResponseGetServiceName

> otError otDnsServiceResponseGetServiceName (const otDnsServiceResponse *aResponse, char *aLabelBuffer, uint8_t aLabelBufferSize, char *aNameBuffer, uint16_t aNameBufferSize)

Gets the service instance name associated with a DNS service instance resolution response.

Parameters

[in]	aResponse	A pointer to the response.

[out]	aLabelBuffer	A buffer to char array to output the service instance label (MUST NOT be NULL).
[in]	aLabelBufferSize	The size of `aLabelBuffer`.
[out]	aNameBuffer	A buffer to char array to output the rest of service name (can be NULL if user is not interested in getting the name.
[in]	aNameBufferSize	The size of `aNameBuffer`.

MUST only be used from `otDnsServiceCallback`.

Definition at line `610` of file `include/openthread/dns_client.h`

otDnsServiceResponseGetServiceInfo

> otError otDnsServiceResponseGetServiceInfo (const otDnsServiceResponse *aResponse, otDnsServiceInfo *aServiceInfo)

Gets info for a service instance from a DNS service instance resolution response.

Parameters

[in]	aResponse	A pointer to the response.
[out]	aServiceInfo	A `ServiceInfo` to output the service instance information (MUST NOT be NULL).

MUST only be used from a `otDnsServiceCallback` triggered from `otDnsClientResolveService()` or `otDnsClientResolveServiceAndHostAddress()`.

When this is is used from a `otDnsClientResolveService()` callback, the DNS response from server/resolver may include AAAA records in its Additional Data section for the host name associated with the service instance that is resolved. This is a SHOULD and not a MUST requirement so servers/resolvers are not required to provide this. This function attempts to parse AAAA record(s) if included in the response. If it is not included `mHostAddress` is set to all zeros (unspecified address). To also resolve the host address, user can use the DNS client API function `otDnsClientResolveServiceAndHostAddress()` which will perform service resolution followed up by a host name address resolution query (when AAAA records are not provided by server/resolver in the SRV query response).

- If a matching SRV record is found in `aResponse`, `aServiceInfo` is updated.
- If no matching SRV record is found, `OT_ERROR_NOT_FOUND` is returned unless the query config for this query used `OT_DNS_SERVICE_MODE_TXT` for `mServiceMode` (meaning the request was only for TXT record). In this case, we still try to parse the SRV record from Additional Data Section of response (in case server provided the info).
- If no matching TXT record is found in `aResponse`, `mTxtDataSize` in `aServiceInfo` is set to zero.
- If TXT data length is greater than `mTxtDataSize`, it is read partially and `mTxtDataTruncated` is set to true.
- If no matching AAAA record is found in `aResponse`, `mHostAddress is set to all zero or unspecified address.`
- `If there are multiple AAAA records for the host name in @p aResponse,` mHostAddress `is set to the first one. The other addresses can be retrieved using` otDnsServiceResponseGetHostAddress()`.

Definition at line `649` of file `include/openthread/dns_client.h`

otDnsServiceResponseGetHostAddress

> otError otDnsServiceResponseGetHostAddress (const otDnsServiceResponse *aResponse, const char *aHostName, uint16_t aIndex, otIp6Address *aAddress, uint32_t *aTtl)

Gets the host IPv6 address from a DNS service instance resolution response.

Parameters

[in]	aResponse	A pointer to the response.

[in]	aHostName	The host name to get the address (MUST NOT be NULL).
[in]	aIndex	The address record index to retrieve.
[out]	aAddress	A pointer to a IPv6 address to output the address (MUST NOT be NULL).
[out]	aTtl	A pointer to an `uint32_t` to output TTL for the address. It can be NULL if caller does not want to get the TTL.

MUST only be used from `otDnsServiceCallback`.

The response can include zero or more IPv6 address records. `aIndex` can be used to iterate through the list of addresses. Index zero gets the first address and so on. When we reach end of the list, `OT_ERROR_NOT_FOUND` is returned.

Definition at line `672` of file `include/openthread/dns_client.h`

Macro Definition Documentation

OT_DNS_MAX_NAME_SIZE

```
#define OT_DNS_MAX_NAME_SIZE
```

Value:

```
255
```

Maximum name string size (includes null char at the end of string).

Definition at line `57` of file `include/openthread/dns.h`

OT_DNS_MAX_LABEL_SIZE

```
#define OT_DNS_MAX_LABEL_SIZE
```

Value:

```
64
```

Maximum label string size (include null char at the end of string).

Definition at line `59` of file `include/openthread/dns.h`

OT_DNS_TXT_KEY_MIN_LENGTH

```
#define OT_DNS_TXT_KEY_MIN_LENGTH
```

Value:

```
1
```

Minimum length of TXT record key string (RFC 6763 - section 6.4).

Definition at line `61` of file `include/openthread/dns.h`

OT_DNS_TXT_KEY_MAX_LENGTH

```
#define OT_DNS_TXT_KEY_MAX_LENGTH
```

Value:

```
9
```

Recommended maximum length of TXT record key string (RFC 6763 - section 6.4).

Definition at line 63 of file include/openthread/dns.h

otDnsTxtEntry

Represents a TXT record entry representing a key/value pair (RFC 6763 - section 6.3).

The string buffers pointed to by `mKey` and `mValue` MUST persist and remain unchanged after an instance of such structure is passed to OpenThread (as part of `otSrpClientService` instance).

An array of `otDnsTxtEntry` entries are used in `otSrpClientService` to specify the full TXT record (a list of entries).

Public Attributes

const char *	mKey The TXT record key string.
const uint8_t *	mValue The TXT record value or already encoded TXT-DATA (depending on `mKey`).
uint16_t	mValueLength Number of bytes in `mValue` buffer.

Public Attribute Documentation

mKey

```
const char* otDnsTxtEntry::mKey
```

The TXT record key string.

If `mKey` is not NULL, then it MUST be a null-terminated C string. The entry is treated as key/value pair with `mValue` buffer providing the value.

- The entry is encoded as follows:
  - A single string length byte followed by "key=value" format (without the quotation marks).
  - In this case, the overall encoded length must be 255 bytes or less.
- If `mValue` is NULL, then key is treated as a boolean attribute and encoded as "key" (with no `=`).
- If `mValue` is not NULL but `mValueLength` is zero, then it is treated as empty value and encoded as "key=".

If `mKey` is NULL, then `mValue` buffer is treated as an already encoded TXT-DATA and is appended as is in the DNS message.

Definition at line `92` of file `include/openthread/dns.h`

mValue

```
const uint8_t* otDnsTxtEntry::mValue
```

The TXT record value or already encoded TXT-DATA (depending on `mKey`).

Definition at line `93` of file `include/openthread/dns.h`

mValueLength

```
uint16_t otDnsTxtEntry::mValueLength
```

Number of bytes in `mValue` buffer.

Definition at line `94` of file `include/openthread/dns.h`

otDnsTxtEntryIterator

Represents an iterator for TXT record entries (key/value pairs).

The data fields in this structure are intended for use by OpenThread core and caller should not read or change them.

Public Attributes

const void *	mPtr
uint16_t	mData
char	mChar

Public Attribute Documentation

mPtr

```
const void* otDnsTxtEntryIterator::mPtr
```

Definition at line 105 of file include/openthread/dns.h

mData

```
uint16_t otDnsTxtEntryIterator::mData[2]
```

Definition at line 106 of file include/openthread/dns.h

mChar

```
char otDnsTxtEntryIterator::mChar[OT_DNS_TXT_KEY_MAX_LENGTH+1]
```

Definition at line 107 of file include/openthread/dns.h

otDnsQueryConfig

Represents a DNS query configuration.

Any of the fields in this structure can be set to zero to indicate that it is not specified. How the unspecified fields are treated is determined by the function which uses the instance of `otDnsQueryConfig`.

Public Attributes

otSockAddr	mServerSockAddr Server address (IPv6 addr/port). All zero or zero port for unspecified.
uint32_t	mResponseTimeout Wait time (in msec) to rx response. Zero indicates unspecified value.
uint8_t	mMaxTxAttempts Maximum tx attempts before reporting failure. Zero for unspecified value.
otDnsRecursionFlag	mRecursionFlag Indicates whether the server can resolve the query recursively or not.
otDnsNat64Mode	mNat64Mode Allow/Disallow NAT64 address translation during address resolution.
otDnsServiceMode	mServiceMode Determines which records to query during service resolution.
otDnsTransportProto	mTransportProto Select default transport protocol.

Public Attribute Documentation

mServerSockAddr

> otSockAddr otDnsQueryConfig::mServerSockAddr

Server address (IPv6 addr/port). All zero or zero port for unspecified.

Definition at line `122` of file `include/openthread/dns_client.h`

mResponseTimeout

> uint32_t otDnsQueryConfig::mResponseTimeout

Wait time (in msec) to rx response. Zero indicates unspecified value.

Definition at line `123` of file `include/openthread/dns_client.h`

mMaxTxAttempts

uint8_t otDnsQueryConfig::mMaxTxAttempts

Maximum tx attempts before reporting failure. Zero for unspecified value.

Definition at line 124 of file include/openthread/dns_client.h

mRecursionFlag

otDnsRecursionFlag otDnsQueryConfig::mRecursionFlag

Indicates whether the server can resolve the query recursively or not.

Definition at line 125 of file include/openthread/dns_client.h

mNat64Mode

otDnsNat64Mode otDnsQueryConfig::mNat64Mode

Allow/Disallow NAT64 address translation during address resolution.

Definition at line 126 of file include/openthread/dns_client.h

mServiceMode

otDnsServiceMode otDnsQueryConfig::mServiceMode

Determines which records to query during service resolution.

Definition at line 127 of file include/openthread/dns_client.h

mTransportProto

otDnsTransportProto otDnsQueryConfig::mTransportProto

Select default transport protocol.

Definition at line 128 of file include/openthread/dns_client.h

otDnsServiceInfo

Provides info for a DNS service instance.

Public Attributes

uint32_t	mTtl Service record TTL (in seconds).
uint16_t	mPort Service port number.
uint16_t	mPriority Service priority.
uint16_t	mWeight Service weight.
char *	mHostNameBuffer Buffer to output the service host name (can be NULL if not needed).
uint16_t	mHostNameBufferSize Size of `mHostNameBuffer`.
otIp6Address	mHostAddress The host IPv6 address. Set to all zero if not available.
uint32_t	mHostAddressTtl The host address TTL.
uint8_t *	mTxtData Buffer to output TXT data (can be NULL if not needed).
uint16_t	mTxtDataSize On input, size of `mTxtData` buffer. On output number bytes written.
bool	mTxtDataTruncated Indicates if TXT data could not fit in `mTxtDataSize` and was truncated.
uint32_t	mTxtDataTtl The TXT data TTL.

Public Attribute Documentation

mTtl

```
uint32_t otDnsServiceInfo::mTtl
```

Service record TTL (in seconds).

Definition at line `349` of file `include/openthread/dns_client.h`

mPort

uint16_t otDnsServiceInfo::mPort

Service port number.

Definition at line `350` of file `include/openthread/dns_client.h`

mPriority

uint16_t otDnsServiceInfo::mPriority

Service priority.

Definition at line `351` of file `include/openthread/dns_client.h`

mWeight

uint16_t otDnsServiceInfo::mWeight

Service weight.

Definition at line `352` of file `include/openthread/dns_client.h`

mHostNameBuffer

char* otDnsServiceInfo::mHostNameBuffer

Buffer to output the service host name (can be NULL if not needed).

Definition at line `353` of file `include/openthread/dns_client.h`

mHostNameBufferSize

uint16_t otDnsServiceInfo::mHostNameBufferSize

Size of `mHostNameBuffer`.

Definition at line `354` of file `include/openthread/dns_client.h`

mHostAddress

otIp6Address otDnsServiceInfo::mHostAddress

The host IPv6 address. Set to all zero if not available.

Definition at line `355` of file `include/openthread/dns_client.h`

mHostAddressTtl

uint32_t otDnsServiceInfo::mHostAddressTtl

The host address TTL.

Definition at line 356 of file include/openthread/dns_client.h

mTxtData

uint8_t* otDnsServiceInfo::mTxtData

Buffer to output TXT data (can be NULL if not needed).

Definition at line 357 of file include/openthread/dns_client.h

mTxtDataSize

uint16_t otDnsServiceInfo::mTxtDataSize

On input, size of mTxtData buffer. On output number bytes written.

Definition at line 358 of file include/openthread/dns_client.h

mTxtDataTruncated

bool otDnsServiceInfo::mTxtDataTruncated

Indicates if TXT data could not fit in mTxtDataSize and was truncated.

Definition at line 359 of file include/openthread/dns_client.h

mTxtDataTtl

uint32_t otDnsServiceInfo::mTxtDataTtl

The TXT data TTL.

Definition at line 360 of file include/openthread/dns_client.h

DNS-SD Server

DNS-SD Server

This module includes APIs for DNS-SD server.

Modules

otDnssdServiceInstanceInfo

otDnssdHostInfo

otDnssdCounters

Enumerations

enum otDnssdQueryType {

OT_DNSSD_QUERY_TYPE_NONE = 0
OT_DNSSD_QUERY_TYPE_BROWSE = 1
OT_DNSSD_QUERY_TYPE_RESOLVE = 2
OT_DNSSD_QUERY_TYPE_RESOLVE_HOST = 3

}

Specifies a DNS-SD query type.

Typedefs

typedef void(* otDnssdQuerySubscribeCallback)(void *aContext, const char *aFullName)

Is called when a DNS-SD query subscribes one of:

typedef void(* otDnssdQueryUnsubscribeCallback)(void *aContext, const char *aFullName)

Is called when a DNS-SD query unsubscribes one of:

typedef void otDnssdQuery

This opaque type represents a DNS-SD query.

typedef struct otDnssdServiceInstanceInfo otDnssdServiceInstanceInfo

Represents information of a discovered service instance for a DNS-SD query.

typedef struct otDnssdHostInfo otDnssdHostInfo

Represents information of a discovered host for a DNS-SD query.

typedef struct otDnssdCounters otDnssdCounters

Contains the counters of DNS-SD server.

Functions

void otDnssdQuerySetCallbacks(otInstance *aInstance, otDnssdQuerySubscribeCallback aSubscribe, otDnssdQueryUnsubscribeCallback aUnsubscribe, void *aContext)

Sets DNS-SD server query callbacks.

void	otDnssdQueryHandleDiscoveredServiceInstance(otInstance *aInstance, const char *aServiceFullName, otDnssdServiceInstanceInfo *aInstanceInfo) Notifies a discovered service instance.
void	otDnssdQueryHandleDiscoveredHost(otInstance *aInstance, const char *aHostFullName, otDnssdHostInfo *aHostInfo) Notifies a discovered host.
const otDnssdQuery *	otDnssdGetNextQuery(otInstance *aInstance, const otDnssdQuery *aQuery) Acquires the next query in the DNS-SD server.
otDnssdQueryType	otDnssdGetQueryTypeAndName(const otDnssdQuery *aQuery, char(*aNameOutput)[OT_DNS_MAX_NAME_SIZE]) Acquires the DNS-SD query type and name for a specific query.
const otDnssdCounters *	otDnssdGetCounters(otInstance *aInstance) Returns the counters of the DNS-SD server.
void	otDnssdUpstreamQuerySetEnabled(otInstance *aInstance, bool aEnabled) Enable or disable forwarding DNS queries to platform DNS upstream API.
bool	otDnssdUpstreamQueryIsEnabled(otInstance *aInstance) Returns whether the DNSSD server will forward DNS queries to the platform DNS upstream API.

Enumeration Documentation

otDnssdQueryType

> otDnssdQueryType

Specifies a DNS-SD query type.

Enumerator	
OT_DNSSD_QUERY_TYPE_NONE	Service type unspecified.
OT_DNSSD_QUERY_TYPE_BROWSE	Service type browse service.
OT_DNSSD_QUERY_TYPE_RESOLVE	Service type resolve service instance.
OT_DNSSD_QUERY_TYPE_RESOLVE_HOST	Service type resolve hostname.

Definition at line 142 of file include/openthread/dnssd_server.h

Typedef Documentation

otDnssdQuerySubscribeCallback

> typedef void(* otDnssdQuerySubscribeCallback) (void *aContext, const char *aFullName))(void *aContext, const char *aFullName)

Is called when a DNS-SD query subscribes one of:

Parameters

[in]	aContext	A pointer to the application-specific context.
[in]	aFullName	The null-terminated full service name (e.g. "_ipps._tcp.default.service.arpa."), or full service instance name (e.g. "OpenThread._ipps._tcp.default.service.arpa."), or full host name (e.g. "ot-host.default.service.arpa.").

1. a service name.
2. a service instance name.

a host name.

The DNS-SD query implementation is responsible for identifying what `aFullName` is. If `aFullName` is a service name or service instance name, the DNS-SD query implementation should discover corresponding service instance information and notify the DNS-SD server using `otDnssdQueryHandleDiscoveredServiceInstance`. If `aFullName` is a host name, the DNS-SD query implementation should discover the host information and notify the DNS-SD server using `otDnssdQueryHandleDiscoveredHost`.

Note

- There can be multiple subscription to the same name. DNS-SD query implementation should record the number of active subscriptions and stop notifying when there is no active subscription for `aFullName`.

See Also

- otDnssdQueryHandleDiscoveredServiceInstance
- otDnssdQueryHandleDiscoveredHost

Definition at line `83` of file `include/openthread/dnssd_server.h`

otDnssdQueryUnsubscribeCallback

```
typedef void(* otDnssdQueryUnsubscribeCallback) (void *aContext, const char *aFullName) )(void *aContext, const char *aFullName)
```

Is called when a DNS-SD query unsubscribes one of:

Parameters

[in]	aContext	A pointer to the application-specific context.
[in]	aFullName	The null-terminated full service name (e.g. "_ipps._tcp.default.service.arpa."), or full service instance name (e.g. "OpenThread._ipps._tcp.default.service.arpa.").

1. a service name.
2. a service instance name.
3. a host name.

The DNS-SD query implementation is responsible for identifying what `aFullName` is.

Note

- There can be multiple subscription to the same name. DNS-SD query implementation should record the number of active subscriptions and stop notifying when there is no active subscription for `aFullName`.

Definition at line `101` of file `include/openthread/dnssd_server.h`

otDnssdQuery

```
typedef void otDnssdQuery
```

This opaque type represents a DNS-SD query.

Definition at line `107` of file `include/openthread/dnssd_server.h`

otDnssdServiceInstanceInfo

```
typedef struct otDnssdServiceInstanceInfo otDnssdServiceInstanceInfo
```

Represents information of a discovered service instance for a DNS-SD query.

Definition at line 125 of file include/openthread/dnssd_server.h

otDnssdHostInfo

```
typedef struct otDnssdHostInfo otDnssdHostInfo
```

Represents information of a discovered host for a DNS-SD query.

Definition at line 136 of file include/openthread/dnssd_server.h

otDnssdCounters

```
typedef struct otDnssdCounters otDnssdCounters
```

Contains the counters of DNS-SD server.

Definition at line 164 of file include/openthread/dnssd_server.h

Function Documentation

otDnssdQuerySetCallbacks

```
void otDnssdQuerySetCallbacks (otInstance *aInstance, otDnssdQuerySubscribeCallback aSubscribe,
otDnssdQueryUnsubscribeCallback aUnsubscribe, void *aContext)
```

Sets DNS-SD server query callbacks.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aSubscribe	A pointer to the callback function to subscribe a service or service instance.
[in]	aUnsubscribe	A pointer to the callback function to unsubscribe a service or service instance.
[in]	aContext	A pointer to the application-specific context.

The DNS-SD server calls `aSubscribe` to subscribe to a service or service instance to resolve a DNS-SD query and `aUnsubscribe` to unsubscribe when the query is resolved or timeout.

Note

- `aSubscribe` and `aUnsubscribe` must be both set or unset.

Definition at line 180 of file include/openthread/dnssd_server.h

otDnssdQueryHandleDiscoveredServiceInstance

```
void otDnssdQueryHandleDiscoveredServiceInstance (otInstance *aInstance, const char *aServiceFullName,
otDnssdServiceInstanceInfo *aInstanceInfo)
```

Notifies a discovered service instance.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aServiceFullName	The null-terminated full service name.
[in]	aInstanceInfo	A pointer to the discovered service instance information.

The external query resolver (e.g. Discovery Proxy) should call this function to notify OpenThread core of the subscribed services or service instances.

Note

- `aInstanceInfo` must not contain unspecified or link-local or loop-back or multicast IP addresses.

Definition at line `198` of file `include/openthread/dnssd_server.h`

otDnssdQueryHandleDiscoveredHost

```
void otDnssdQueryHandleDiscoveredHost (otInstance *aInstance, const char *aHostFullName, otDnssdHostInfo *aHostInfo)
```

Notifies a discovered host.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aHostFullName	The null-terminated full host name.
[in]	aHostInfo	A pointer to the discovered service instance information.

The external query resolver (e.g. Discovery Proxy) should call this function to notify OpenThread core of the subscribed hosts.

Note

- `aHostInfo` must not contain unspecified or link-local or loop-back or multicast IP addresses.

Definition at line `214` of file `include/openthread/dnssd_server.h`

otDnssdGetNextQuery

```
const otDnssdQuery * otDnssdGetNextQuery (otInstance *aInstance, const otDnssdQuery *aQuery)
```

Acquires the next query in the DNS-SD server.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aQuery	The query pointer. Pass NULL to get the first query.

Returns

- A pointer to the query or NULL if no more queries.

Definition at line `225` of file `include/openthread/dnssd_server.h`

otDnssdGetQueryTypeAndName

```
otDnssdQueryType otDnssdGetQueryTypeAndName (const otDnssdQuery *aQuery, char(*aNameOutput)[OT_DNS_MAX_NAME_SIZE])
```

Acquires the DNS-SD query type and name for a specific query.

Parameters

[in]	aQuery	The query pointer acquired from `otDnssdGetNextQuery`.
[out]	aNameOutput	The name output buffer, which should be `OT_DNS_MAX_NAME_SIZE` bytes long.

Returns

- The DNS-SD query type.

Definition at line `236` of file `include/openthread/dnssd_server.h`

otDnssdGetCounters

```
const otDnssdCounters * otDnssdGetCounters (otInstance *aInstance)
```

Returns the counters of the DNS-SD server.

Parameters

[in]	aInstance	The OpenThread instance structure.

Returns

- A pointer to the counters of the DNS-SD server.

Definition at line `246` of file `include/openthread/dnssd_server.h`

otDnssdUpstreamQuerySetEnabled

```
void otDnssdUpstreamQuerySetEnabled (otInstance *aInstance, bool aEnabled)
```

Enable or disable forwarding DNS queries to platform DNS upstream API.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	A boolean to enable/disable forwarding DNS queries to upstream.

Available when `OPENTHREAD_CONFIG_DNS_UPSTREAM_QUERY_ENABLE` is enabled.

See Also

- otPlatDnsStartUpstreamQuery
- otPlatDnsCancelUpstreamQuery
- otPlatDnsUpstreamQueryDone

Definition at line `261` of file `include/openthread/dnssd_server.h`

otDnssdUpstreamQueryIsEnabled

```
bool otDnssdUpstreamQueryIsEnabled (otInstance *aInstance)
```

Returns whether the DNSSD server will forward DNS queries to the platform DNS upstream API.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Available when `OPENTHREAD_CONFIG_DNS_UPSTREAM_QUERY_ENABLE` is enabled.

See Also

- otDnssdUpstreamQuerySetEnabled

Definition at line `275` of file `include/openthread/dnssd_server.h`

otDnssdServiceInstanceInfo

Represents information of a discovered service instance for a DNS-SD query.

Public Attributes

const char *	mFullName Full instance name (e.g. "OpenThread._ipps._tcp.default.service.arpa.").
const char *	mHostName Host name (e.g. "ot-host.default.service.arpa.").
uint8_t	mAddressNum Number of host IPv6 addresses.
const otIp6Address *	mAddresses Host IPv6 addresses.
uint16_t	mPort Service port.
uint16_t	mPriority Service priority.
uint16_t	mWeight Service weight.
uint16_t	mTxtLength Service TXT RDATA length.
const uint8_t *	mTxtData Service TXT RDATA.
uint32_t	mTtl Service TTL (in seconds).

Public Attribute Documentation

mFullName

```
const char* otDnssdServiceInstanceInfo::mFullName
```

Full instance name (e.g. "OpenThread._ipps._tcp.default.service.arpa.").

Definition at line 115 of file include/openthread/dnssd_server.h

mHostName

```
const char* otDnssdServiceInstanceInfo::mHostName
```

Host name (e.g. "ot-host.default.service.arpa.").

Definition at line 116 of file include/openthread/dnssd_server.h

mAddressNum

```
uint8_t otDnssdServiceInstanceInfo::mAddressNum
```

Number of host IPv6 addresses.

Definition at line 117 of file include/openthread/dnssd_server.h

mAddresses

```
const otIp6Address* otDnssdServiceInstanceInfo::mAddresses
```

Host IPv6 addresses.

Definition at line 118 of file include/openthread/dnssd_server.h

mPort

```
uint16_t otDnssdServiceInstanceInfo::mPort
```

Service port.

Definition at line 119 of file include/openthread/dnssd_server.h

mPriority

```
uint16_t otDnssdServiceInstanceInfo::mPriority
```

Service priority.

Definition at line 120 of file include/openthread/dnssd_server.h

mWeight

```
uint16_t otDnssdServiceInstanceInfo::mWeight
```

Service weight.

Definition at line 121 of file include/openthread/dnssd_server.h

mTxtLength

```
uint16_t otDnssdServiceInstanceInfo::mTxtLength
```

Service TXT RDATA length.

Definition at line 122 of file include/openthread/dnssd_server.h

mTxtData

```
const uint8_t* otDnssdServiceInstanceInfo::mTxtData
```

Service TXT RDATA.

Definition at line 123 of file include/openthread/dnssd_server.h

mTtl

```
uint32_t otDnssdServiceInstanceInfo::mTtl
```

Service TTL (in seconds).

Definition at line 124 of file include/openthread/dnssd_server.h

otDnssdHostInfo

Represents information of a discovered host for a DNS-SD query.

Public Attributes

uint8_t	mAddressNum Number of host IPv6 addresses.
const otIp6Address *	mAddresses Host IPv6 addresses.
uint32_t	mTtl Service TTL (in seconds).

Public Attribute Documentation

mAddressNum

```
uint8_t otDnssdHostInfo::mAddressNum
```

Number of host IPv6 addresses.

Definition at line 133 of file include/openthread/dnssd_server.h

mAddresses

```
const otIp6Address* otDnssdHostInfo::mAddresses
```

Host IPv6 addresses.

Definition at line 134 of file include/openthread/dnssd_server.h

mTtl

```
uint32_t otDnssdHostInfo::mTtl
```

Service TTL (in seconds).

Definition at line 135 of file include/openthread/dnssd_server.h

otDnssdCounters

Contains the counters of DNS-SD server.

Public Attributes

uint32_t	mSuccessResponse The number of successful responses.
uint32_t	mServerFailureResponse The number of server failure responses.
uint32_t	mFormatErrorResponse The number of format error responses.
uint32_t	mNameErrorResponse The number of name error responses.
uint32_t	mNotImplementedResponse The number of 'not implemented' responses.
uint32_t	mOtherResponse The number of other responses.
uint32_t	mResolvedBySrp The number of queries completely resolved by the local SRP server.

Public Attribute Documentation

mSuccessResponse

```
uint32_t otDnssdCounters::mSuccessResponse
```

The number of successful responses.

Definition at line 156 of file include/openthread/dnssd_server.h

mServerFailureResponse

```
uint32_t otDnssdCounters::mServerFailureResponse
```

The number of server failure responses.

Definition at line 157 of file include/openthread/dnssd_server.h

mFormatErrorResponse

```
uint32_t otDnssdCounters::mFormatErrorResponse
```

The number of format error responses.

Definition at line 158 of file include/openthread/dnssd_server.h

mNameErrorResponse

```
uint32_t otDnssdCounters::mNameErrorResponse
```

The number of name error responses.

Definition at line 159 of file include/openthread/dnssd_server.h

mNotImplementedResponse

```
uint32_t otDnssdCounters::mNotImplementedResponse
```

The number of 'not implemented' responses.

Definition at line 160 of file include/openthread/dnssd_server.h

mOtherResponse

```
uint32_t otDnssdCounters::mOtherResponse
```

The number of other responses.

Definition at line 161 of file include/openthread/dnssd_server.h

mResolvedBySrp

```
uint32_t otDnssdCounters::mResolvedBySrp
```

The number of queries completely resolved by the local SRP server.

Definition at line 163 of file include/openthread/dnssd_server.h

ICMPv6

ICMPv6

This module includes functions that control ICMPv6 communication.

Modules

otIcmp6Header

otIcmp6Handler

Enumerations

enum otIcmp6Type {

OT_ICMP6_TYPE_DST_UNREACH = 1
OT_ICMP6_TYPE_PACKET_TO_BIG = 2
OT_ICMP6_TYPE_TIME_EXCEEDED = 3
OT_ICMP6_TYPE_PARAMETER_PROBLEM = 4
OT_ICMP6_TYPE_ECHO_REQUEST = 128
OT_ICMP6_TYPE_ECHO_REPLY = 129
OT_ICMP6_TYPE_ROUTER_SOLICIT = 133
OT_ICMP6_TYPE_ROUTER_ADVERT = 134
OT_ICMP6_TYPE_NEIGHBOR_SOLICIT = 135
OT_ICMP6_TYPE_NEIGHBOR_ADVERT = 136

}

ICMPv6 Message Types.

enum otIcmp6Code {

OT_ICMP6_CODE_DST_UNREACH_NO_ROUTE = 0
OT_ICMP6_CODE_FRAGM_REAS_TIME_EX = 1

}

ICMPv6 Message Codes.

enum otIcmp6EchoMode {

OT_ICMP6_ECHO_HANDLER_DISABLED = 0
OT_ICMP6_ECHO_HANDLER_UNICAST_ONLY = 1
OT_ICMP6_ECHO_HANDLER_MULTICAST_ONLY = 2
OT_ICMP6_ECHO_HANDLER_ALL = 3

}

ICMPv6 Echo Reply Modes.

Typedefs

typedef enum otIcmp6Type — otIcmp6Type
ICMPv6 Message Types.

typedef enum otIcmp6Code — otIcmp6Code
ICMPv6 Message Codes.

typedef struct otIcmp6Header — otIcmp6Header
Represents an ICMPv6 header.

typedef void(*	otIcmp6ReceiveCallback)(void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo, const otIcmp6Header *aIcmpHeader) This callback allows OpenThread to inform the application of a received ICMPv6 message.
typedef struct otIcmp6Handler	otIcmp6Handler Implements ICMPv6 message handler.
typedef enum otIcmp6EchoMode	otIcmp6EchoMode ICMPv6 Echo Reply Modes.

Variables

OT_TOOL_PACKED_BEGIN struct otIcmp6Header	OT_TOOL_PACKED_END

Functions

otIcmp6EchoMode	otIcmp6GetEchoMode(otInstance *aInstance) Indicates whether or not ICMPv6 Echo processing is enabled.
void	otIcmp6SetEchoMode(otInstance *aInstance, otIcmp6EchoMode aMode) Sets whether or not ICMPv6 Echo processing is enabled.
otError	otIcmp6RegisterHandler(otInstance *aInstance, otIcmp6Handler *aHandler) Registers a handler to provide received ICMPv6 messages.
otError	otIcmp6SendEchoRequest(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, uint16_t aIdentifier) Sends an ICMPv6 Echo Request via the Thread interface.

Macros

#define	OT_ICMP6_HEADER_DATA_SIZE 4 Size of ICMPv6 Header.
#define	OT_ICMP6_ROUTER_ADVERT_MIN_SIZE 16 Size of a Router Advertisement message without any options.

Enumeration Documentation

otIcmp6Type

> otIcmp6Type

ICMPv6 Message Types.

Enumerator	
OT_ICMP6_TYPE_DST_UNREACH	Destination Unreachable.
OT_ICMP6_TYPE_PACKET_TO_BIG	Packet To Big.
OT_ICMP6_TYPE_TIME_EXCEEDED	Time Exceeded.
OT_ICMP6_TYPE_PARAMETER_PROBLEM	Parameter Problem.
OT_ICMP6_TYPE_ECHO_REQUEST	Echo Request.
OT_ICMP6_TYPE_ECHO_REPLY	Echo Reply.

OT_ICMP6_TYPE_ROUTER_SOLICIT	Router Solicitation.
OT_ICMP6_TYPE_ROUTER_ADVERT	Router Advertisement.
OT_ICMP6_TYPE_NEIGHBOR_SOLICIT	Neighbor Solicitation.
OT_ICMP6_TYPE_NEIGHBOR_ADVERT	Neighbor Advertisement.

Definition at line 59 of file include/openthread/icmp6.h

otIcmp6Code

> otIcmp6Code

ICMPv6 Message Codes.

Enumerator	
OT_ICMP6_CODE_DST_UNREACH_NO_ROUTE	Destination Unreachable No Route.
OT_ICMP6_CODE_FRAGM_REAS_TIME_EX	Fragment Reassembly Time Exceeded.

Definition at line 77 of file include/openthread/icmp6.h

otIcmp6EchoMode

> otIcmp6EchoMode

ICMPv6 Echo Reply Modes.

Enumerator	
OT_ICMP6_ECHO_HANDLER_DISABLED	ICMPv6 Echo processing disabled.
OT_ICMP6_ECHO_HANDLER_UNICAST_ONLY	ICMPv6 Echo processing enabled only for unicast requests only.
OT_ICMP6_ECHO_HANDLER_MULTICAST_ONLY	ICMPv6 Echo processing enabled only for multicast requests only.
OT_ICMP6_ECHO_HANDLER_ALL	ICMPv6 Echo processing enabled for unicast and multicast requests.

Definition at line 141 of file include/openthread/icmp6.h

Typedef Documentation

otIcmp6Type

> typedef enum otIcmp6Type otIcmp6Type

ICMPv6 Message Types.

Definition at line 71 of file include/openthread/icmp6.h

otIcmp6Code

> typedef enum otIcmp6Code otIcmp6Code

ICMPv6 Message Codes.

Definition at line 81 of file include/openthread/icmp6.h

otIcmp6Header

```
typedef struct otIcmp6Header otIcmp6Header
```

Represents an ICMPv6 header.

Definition at line `110` of file `include/openthread/icmp6.h`

otIcmp6ReceiveCallback

```
typedef void(* otIcmp6ReceiveCallback) (void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo,
const otIcmp6Header *aIcmpHeader) )(void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo,
const otIcmp6Header *aIcmpHeader)
```

This callback allows OpenThread to inform the application of a received ICMPv6 message.

Parameters

[in]	aContext	A pointer to arbitrary context information.
[in]	aMessage	A pointer to the received message.
[in]	aMessageInfo	A pointer to message information associated with `aMessage`.
[in]	aIcmpHeader	A pointer to the received ICMPv6 header.

Definition at line `121` of file `include/openthread/icmp6.h`

otIcmp6Handler

```
typedef struct otIcmp6Handler otIcmp6Handler
```

Implements ICMPv6 message handler.

Definition at line `135` of file `include/openthread/icmp6.h`

otIcmp6EchoMode

```
typedef enum otIcmp6EchoMode otIcmp6EchoMode
```

ICMPv6 Echo Reply Modes.

Definition at line `147` of file `include/openthread/icmp6.h`

Variable Documentation

OT_TOOL_PACKED_END

```
OT_TOOL_PACKED_BEGIN struct otIcmp6Header OT_TOOL_PACKED_END
```

Definition at line `104` of file `include/openthread/icmp6.h`

Function Documentation

otIcmp6GetEchoMode

```
otIcmp6EchoMode otIcmp6GetEchoMode (otInstance *aInstance)
```

Indicates whether or not ICMPv6 Echo processing is enabled.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `160` of file `include/openthread/icmp6.h`

otIcmp6SetEchoMode

```
void otIcmp6SetEchoMode (otInstance *aInstance, otIcmp6EchoMode aMode)
```

Sets whether or not ICMPv6 Echo processing is enabled.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMode	The ICMPv6 Echo processing mode.

Definition at line `169` of file `include/openthread/icmp6.h`

otIcmp6RegisterHandler

```
otError otIcmp6RegisterHandler (otInstance *aInstance, otIcmp6Handler *aHandler)
```

Registers a handler to provide received ICMPv6 messages.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aHandler	A pointer to a handler containing callback that is called when an ICMPv6 message is received.

Note

- A handler structure `aHandler` has to be stored in persistent (static) memory. OpenThread does not make a copy of handler structure.

Definition at line `182` of file `include/openthread/icmp6.h`

otIcmp6SendEchoRequest

```
otError otIcmp6SendEchoRequest (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo,
uint16_t aIdentifier)
```

Sends an ICMPv6 Echo Request via the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[in]	aMessage	A pointer to the message buffer containing the ICMPv6 payload.
[in]	aMessageInfo	A reference to message information associated with `aMessage` .
[in]	aIdentifier	An identifier to aid in matching Echo Replies to this Echo Request. May be zero.

Definition at line `194` of file `include/openthread/icmp6.h`

Macro Definition Documentation

OT_ICMP6_HEADER_DATA_SIZE

```
#define OT_ICMP6_HEADER_DATA_SIZE
```

Value:

```
4
```

Size of ICMPv6 Header.

Definition at line `83` of file `include/openthread/icmp6.h`

OT_ICMP6_ROUTER_ADVERT_MIN_SIZE

```
#define OT_ICMP6_ROUTER_ADVERT_MIN_SIZE
```

Value:

```
16
```

Size of a Router Advertisement message without any options.

Definition at line `84` of file `include/openthread/icmp6.h`

otIcmp6Header

Represents an ICMPv6 header.

Modules

otIcmp6Header::OT_TOOL_PACKED_FIELD

Public Attributes

uint8_t	mType Type.
uint8_t	mCode Code.
uint16_t	mChecksum Checksum.
union otIcmp6Header::OT_TOOL_PACKED_FIELD	mData Message-specific data.

Public Attribute Documentation

mType

```
uint8_t otIcmp6Header::mType
```

Type.

Definition at line 95 of file include/openthread/icmp6.h

mCode

```
uint8_t otIcmp6Header::mCode
```

Code.

Definition at line 96 of file include/openthread/icmp6.h

mChecksum

```
uint16_t otIcmp6Header::mChecksum
```

Checksum.

Definition at line 97 of file include/openthread/icmp6.h

union otIcmp6Header::OT_TOOL_PACKED_FIELD otIcmp6Header::mData

Message-specific data.

Definition at line 103 of file include/openthread/icmp6.h

otIcmp6Header

Public Attributes

uint8_t	m8
uint16_t	m16
uint32_t	m32

Public Attribute Documentation

m8

```
uint8_t otIcmp6Header::OT_TOOL_PACKED_FIELD::m8[OT_ICMP6_HEADER_DATA_SIZE/sizeof(uint8_t)]
```

Definition at line `100` of file `include/openthread/icmp6.h`

m16

```
uint16_t otIcmp6Header::OT_TOOL_PACKED_FIELD::m16[OT_ICMP6_HEADER_DATA_SIZE/sizeof(uint16_t)]
```

Definition at line `101` of file `include/openthread/icmp6.h`

m32

```
uint32_t otIcmp6Header::OT_TOOL_PACKED_FIELD::m32[OT_ICMP6_HEADER_DATA_SIZE/sizeof(uint32_t)]
```

Definition at line `102` of file `include/openthread/icmp6.h`

otIcmp6Handler

Implements ICMPv6 message handler.

Public Attributes

otIcmp6ReceiveCallback	mReceiveCallback The ICMPv6 received callback.
void *	mContext A pointer to arbitrary context information.
struct otIcmp6Handler *	mNext A pointer to the next handler in the list.

Public Attribute Documentation

mReceiveCallback

```
otIcmp6ReceiveCallback otIcmp6Handler::mReceiveCallback
```

The ICMPv6 received callback.

Definition at line 132 of file include/openthread/icmp6.h

mContext

```
void* otIcmp6Handler::mContext
```

A pointer to arbitrary context information.

Definition at line 133 of file include/openthread/icmp6.h

mNext

```
struct otIcmp6Handler* otIcmp6Handler::mNext
```

A pointer to the next handler in the list.

Definition at line 134 of file include/openthread/icmp6.h

IPv6

IPv6

This module includes functions that control IPv6 communication.

Modules

otIp6InterfaceIdentifier

otIp6NetworkPrefix

otIp6AddressComponents

otIp6Address

otIp6Prefix

otNetifAddress

otNetifMulticastAddress

otSockAddr

otMessageInfo

otIp6AddressInfo

otPacketsAndBytes

otBorderRoutingCounters

Enumerations

enum @2 {

OT_ADDRESS_ORIGIN_THREAD = 0
OT_ADDRESS_ORIGIN_SLAAC = 1
OT_ADDRESS_ORIGIN_DHCPV6 = 2
OT_ADDRESS_ORIGIN_MANUAL = 3

}

IPv6 Address origins.

enum @3 {

OT_ECN_NOT_CAPABLE = 0×0
OT_ECN_CAPABLE_0 = 0×2
OT_ECN_CAPABLE_1 = 0×1
OT_ECN_MARKED = 0×3

}

ECN statuses, represented as in the IP header.

enum @4 {

OT_IP6_PROTO_HOP_OPTS = 0
OT_IP6_PROTO_TCP = 6
OT_IP6_PROTO_UDP = 17
OT_IP6_PROTO_IP6 = 41
OT_IP6_PROTO_ROUTING = 43
OT_IP6_PROTO_FRAGMENT = 44

```
    OT_IP6_PROTO_ICMP6 =
    58
    OT_IP6_PROTO_NONE =
    59
    OT_IP6_PROTO_DST_OPTS
    = 60
}
```

Internet Protocol Numbers.

Typedefs

typedef struct otIp6InterfaceIdentifier	otIp6InterfaceIdentifier Represents the Interface Identifier of an IPv6 address.
typedef struct otIp6NetworkPrefix	otIp6NetworkPrefix Represents the Network Prefix of an IPv6 address (most significant 64 bits of the address).
typedef struct otIp6AddressComponents	otIp6AddressComponents Represents the components of an IPv6 address.
typedef struct otIp6Address	otIp6Address Represents an IPv6 address.
typedef struct otIp6Prefix	otIp6Prefix Represents an IPv6 prefix.
typedef struct otNetifAddress	otNetifAddress Represents an IPv6 network interface unicast address.
typedef struct otNetifMulticastAddress	otNetifMulticastAddress Represents an IPv6 network interface multicast address.
typedef struct otSockAddr	otSockAddr Represents an IPv6 socket address.
typedef struct otMessageInfo	otMessageInfo Represents the local and peer IPv6 socket addresses.
typedef void(*	otIp6ReceiveCallback)(otMessage *aMessage, void *aContext) Pointer is called when an IPv6 datagram is received.
typedef struct otIp6AddressInfo	otIp6AddressInfo Represents IPv6 address information.
typedef void(*	otIp6AddressCallback)(const otIp6AddressInfo *aAddressInfo, bool aIsAdded, void *aContext) Pointer is called when an internal IPv6 address is added or removed.
typedef bool(*	otIp6SlaacPrefixFilter)(otInstance *aInstance, const otIp6Prefix *aPrefix) Pointer allows user to filter prefixes and not allow an SLAAC address based on a prefix to be added.
typedef void(*	otIp6RegisterMulticastListenersCallback)(void *aContext, otError aError, uint8_t aMlrStatus, const otIp6Address *aFailedAddresses, uint8_t aFailedAddressNum) Pointer is called with results of `otIp6RegisterMulticastListeners`.
typedef struct otPacketsAndBytes	otPacketsAndBytes Represents the counters for packets and bytes.

typedef struct otBorderRoutingCounters	otBorderRoutingCounters Represents the counters of packets forwarded via Border Routing.

Variables

OT_TOOL_PACKED_BEGIN struct otIp6InterfaceIdentifier	OT_TOOL_PACKED_END

Functions

otError	otIp6SetEnabled(otInstance *aInstance, bool aEnabled) Brings the IPv6 interface up or down.
bool	otIp6IsEnabled(otInstance *aInstance) Indicates whether or not the IPv6 interface is up.
otError	otIp6AddUnicastAddress(otInstance *aInstance, const otNetifAddress *aAddress) Adds a Network Interface Address to the Thread interface.
otError	otIp6RemoveUnicastAddress(otInstance *aInstance, const otIp6Address *aAddress) Removes a Network Interface Address from the Thread interface.
const otNetifAddress *	otIp6GetUnicastAddresses(otInstance *aInstance) Gets the list of IPv6 addresses assigned to the Thread interface.
bool	otIp6HasUnicastAddress(otInstance *aInstance, const otIp6Address *aAddress) Indicates whether or not a unicast IPv6 address is assigned to the Thread interface.
otError	otIp6SubscribeMulticastAddress(otInstance *aInstance, const otIp6Address *aAddress) Subscribes the Thread interface to a Network Interface Multicast Address.
otError	otIp6UnsubscribeMulticastAddress(otInstance *aInstance, const otIp6Address *aAddress) Unsubscribes the Thread interface to a Network Interface Multicast Address.
const otNetifMulticastAddress *	otIp6GetMulticastAddresses(otInstance *aInstance) Gets the list of IPv6 multicast addresses subscribed to the Thread interface.
bool	otIp6IsMulticastPromiscuousEnabled(otInstance *aInstance) Checks if multicast promiscuous mode is enabled on the Thread interface.
void	otIp6SetMulticastPromiscuousEnabled(otInstance *aInstance, bool aEnabled) Enables or disables multicast promiscuous mode on the Thread interface.
otMessage *	otIp6NewMessage(otInstance *aInstance, const otMessageSettings *aSettings) Allocate a new message buffer for sending an IPv6 message.
otMessage *	otIp6NewMessageFromBuffer(otInstance *aInstance, const uint8_t *aData, uint16_t aDataLength, const otMessageSettings *aSettings) Allocate a new message buffer and write the IPv6 datagram to the message buffer for sending an IPv6 message.
void	otIp6SetReceiveCallback(otInstance *aInstance, otIp6ReceiveCallback aCallback, void *aCallbackContext) Registers a callback to provide received IPv6 datagrams.
void	otIp6SetAddressCallback(otInstance *aInstance, otIp6AddressCallback aCallback, void *aCallbackContext) Registers a callback to notify internal IPv6 address changes.

bool otIp6IsReceiveFilterEnabled(otInstance *aInstance)
Indicates whether or not Thread control traffic is filtered out when delivering IPv6 datagrams via the callback specified in otIp6SetReceiveCallback().

void otIp6SetReceiveFilterEnabled(otInstance *aInstance, bool aEnabled)
Sets whether or not Thread control traffic is filtered out when delivering IPv6 datagrams via the callback specified in otIp6SetReceiveCallback().

otError otIp6Send(otInstance *aInstance, otMessage *aMessage)
Sends an IPv6 datagram via the Thread interface.

otError otIp6AddUnsecurePort(otInstance *aInstance, uint16_t aPort)
Adds a port to the allowed unsecured port list.

otError otIp6RemoveUnsecurePort(otInstance *aInstance, uint16_t aPort)
Removes a port from the allowed unsecure port list.

void otIp6RemoveAllUnsecurePorts(otInstance *aInstance)
Removes all ports from the allowed unsecure port list.

const uint16_t * otIp6GetUnsecurePorts(otInstance *aInstance, uint8_t *aNumEntries)
Returns a pointer to the unsecure port list.

bool otIp6IsAddressEqual(const otIp6Address *aFirst, const otIp6Address *aSecond)
Test if two IPv6 addresses are the same.

bool otIp6ArePrefixesEqual(const otIp6Prefix *aFirst, const otIp6Prefix *aSecond)
Test if two IPv6 prefixes are the same.

otError otIp6AddressFromString(const char *aString, otIp6Address *aAddress)
Converts a human-readable IPv6 address string into a binary representation.

otError otIp6PrefixFromString(const char *aString, otIp6Prefix *aPrefix)
Converts a human-readable IPv6 prefix string into a binary representation.

void otIp6AddressToString(const otIp6Address *aAddress, char *aBuffer, uint16_t aSize)
Converts a given IPv6 address to a human-readable string.

void otIp6SockAddrToString(const otSockAddr *aSockAddr, char *aBuffer, uint16_t aSize)
Converts a given IPv6 socket address to a human-readable string.

void otIp6PrefixToString(const otIp6Prefix *aPrefix, char *aBuffer, uint16_t aSize)
Converts a given IPv6 prefix to a human-readable string.

uint8_t otIp6PrefixMatch(const otIp6Address *aFirst, const otIp6Address *aSecond)
Returns the prefix match length (bits) for two IPv6 addresses.

void otIp6GetPrefix(const otIp6Address *aAddress, uint8_t aLength, otIp6Prefix *aPrefix)
Gets a prefix with `aLength` from `aAddress`.

bool otIp6IsAddressUnspecified(const otIp6Address *aAddress)
Indicates whether or not a given IPv6 address is the Unspecified Address.

otError otIp6SelectSourceAddress(otInstance *aInstance, otMessageInfo *aMessageInfo)
Perform OpenThread source address selection.

bool otIp6IsSlaacEnabled(otInstance *aInstance)
Indicates whether the SLAAC module is enabled or not.

void otIp6SetSlaacEnabled(otInstance *aInstance, bool aEnabled)
Enables/disables the SLAAC module.

void	otIp6SetSlaacPrefixFilter(otInstance *aInstance, otIp6SlaacPrefixFilter aFilter) Sets the SLAAC module filter handler.
otError	otIp6RegisterMulticastListeners(otInstance *aInstance, const otIp6Address *aAddresses, uint8_t aAddressNum, const uint32_t *aTimeout, otIp6RegisterMulticastListenersCallback aCallback, void *aContext) Registers Multicast Listeners to Primary Backbone Router.
otError	otIp6SetMeshLocalIid(otInstance *aInstance, const otIp6InterfaceIdentifier *aIid) Sets the Mesh Local IID (for test purpose).
const char *	otIp6ProtoToString(uint8_t aIpProto) Converts a given IP protocol number to a human-readable string.
const otBorderRoutingCounters *	otIp6GetBorderRoutingCounters(otInstance *aInstance) Gets the Border Routing counters.
void	otIp6ResetBorderRoutingCounters(otInstance *aInstance) Resets the Border Routing counters.

Macros

#define	OT_IP6_PREFIX_SIZE 8 Size of an IPv6 prefix (bytes)
#define	OT_IP6_PREFIX_BITSIZE (OT_IP6_PREFIX_SIZE * 8) Size of an IPv6 prefix (bits)
#define	OT_IP6_IID_SIZE 8 Size of an IPv6 Interface Identifier (bytes)
#define	OT_IP6_ADDRESS_SIZE 16 Size of an IPv6 address (bytes)
#define	OT_IP6_HEADER_SIZE 40 Size of an IPv6 header (bytes)
#define	OT_IP6_HEADER_PROTO_OFFSET 6 Offset of the proto field in the IPv6 header (bytes)
#define	OT_IP6_ADDRESS_STRING_SIZE 40 Recommended size for string representation of an IPv6 address.
#define	OT_IP6_SOCK_ADDR_STRING_SIZE 48 Recommended size for string representation of an IPv6 socket address.
#define	OT_IP6_PREFIX_STRING_SIZE 45 Recommended size for string representation of an IPv6 prefix.
#define	OT_IP6_MAX_MLR_ADDRESSES 15 Max number of IPv6 addresses supported by Multicast Listener Registration.

Enumeration Documentation

@2

> @2

IPv6 Address origins.

Enumerator

OT_ADDRESS_ORIGIN_THREAD	Thread assigned address (ALOC, RLOC, MLEID, etc)
OT_ADDRESS_ORIGIN_SLAAC	SLAAC assigned address.
OT_ADDRESS_ORIGIN_DHCPV6	DHCPv6 assigned address.
OT_ADDRESS_ORIGIN_MANUAL	Manually assigned address.

Definition at line 169 of file include/openthread/ip6.h

@3

> @3

ECN statuses, represented as in the IP header.

Enumerator	
OT_ECN_NOT_CAPABLE	Non-ECT.
OT_ECN_CAPABLE_0	ECT(0)
OT_ECN_CAPABLE_1	ECT(1)
OT_ECN_MARKED	Congestion encountered (CE)

Definition at line 219 of file include/openthread/ip6.h

@4

> @4

Internet Protocol Numbers.

Enumerator	
OT_IP6_PROTO_HOP_OPTS	IPv6 Hop-by-Hop Option.
OT_IP6_PROTO_TCP	Transmission Control Protocol.
OT_IP6_PROTO_UDP	User Datagram.
OT_IP6_PROTO_IP6	IPv6 encapsulation.
OT_IP6_PROTO_ROUTING	Routing Header for IPv6.
OT_IP6_PROTO_FRAGMENT	Fragment Header for IPv6.
OT_IP6_PROTO_ICMP6	ICMP for IPv6.
OT_IP6_PROTO_NONE	No Next Header for IPv6.
OT_IP6_PROTO_DST_OPTS	Destination Options for IPv6.

Definition at line 251 of file include/openthread/ip6.h

Typedef Documentation

otIp6InterfaceIdentifier

> typedef struct otIp6InterfaceIdentifier otIp6InterfaceIdentifier

Represents the Interface Identifier of an IPv6 address.

Definition at line 83 of file include/openthread/ip6.h

otIp6NetworkPrefix

```
typedef struct otIp6NetworkPrefix otIp6NetworkPrefix
```

Represents the Network Prefix of an IPv6 address (most significant 64 bits of the address).

Definition at line `101` of file `include/openthread/ip6.h`

otIp6AddressComponents

```
typedef struct otIp6AddressComponents otIp6AddressComponents
```

Represents the components of an IPv6 address.

Definition at line `120` of file `include/openthread/ip6.h`

otIp6Address

```
typedef struct otIp6Address otIp6Address
```

Represents an IPv6 address.

Definition at line `144` of file `include/openthread/ip6.h`

otIp6Prefix

```
typedef struct otIp6Prefix otIp6Prefix
```

Represents an IPv6 prefix.

Definition at line `163` of file `include/openthread/ip6.h`

otNetifAddress

```
typedef struct otNetifAddress otNetifAddress
```

Represents an IPv6 network interface unicast address.

Definition at line `193` of file `include/openthread/ip6.h`

otNetifMulticastAddress

```
typedef struct otNetifMulticastAddress otNetifMulticastAddress
```

Represents an IPv6 network interface multicast address.

Definition at line `203` of file `include/openthread/ip6.h`

otSockAddr

```
typedef struct otSockAddr otSockAddr
```

Represents an IPv6 socket address.

Definition at line 213 of file include/openthread/ip6.h

otMessageInfo

```
typedef struct otMessageInfo otMessageInfo
```

Represents the local and peer IPv6 socket addresses.

Definition at line 245 of file include/openthread/ip6.h

otIp6ReceiveCallback

```
typedef void(* otIp6ReceiveCallback) (otMessage *aMessage, void *aContext) )(otMessage *aMessage, void *aContext)
```

Pointer is called when an IPv6 datagram is received.

Parameters

[in]	aMessage	A pointer to the message buffer containing the received IPv6 datagram. This function transfers the ownership of the `aMessage` to the receiver of the callback. The message should be freed by the receiver of the callback after it is processed (see otMessageFree()).
[in]	aContext	A pointer to application-specific context.

Definition at line 451 of file include/openthread/ip6.h

otIp6AddressInfo

```
typedef struct otIp6AddressInfo otIp6AddressInfo
```

Represents IPv6 address information.

Definition at line 480 of file include/openthread/ip6.h

otIp6AddressCallback

```
typedef void(* otIp6AddressCallback) (const otIp6AddressInfo *aAddressInfo, bool aIsAdded, void *aContext) )(const otIp6AddressInfo *aAddressInfo, bool aIsAdded, void *aContext)
```

Pointer is called when an internal IPv6 address is added or removed.

Parameters

[in]	aAddressInfo	A pointer to the IPv6 address information.
[in]	aIsAdded	TRUE if the `aAddress` was added, FALSE if `aAddress` was removed.
[in]	aContext	A pointer to application-specific context.

Definition at line `490` of file `include/openthread/ip6.h`

otIp6SlaacPrefixFilter

```
typedef bool(* otIp6SlaacPrefixFilter) (otInstance *aInstance, const otIp6Prefix *aPrefix) )(otInstance *aInstance, const otIp6Prefix *aPrefix)
```

Pointer allows user to filter prefixes and not allow an SLAAC address based on a prefix to be added.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPrefix	A pointer to prefix for which SLAAC address is about to be added.

`otIp6SetSlaacPrefixFilter()` can be used to set the filter handler. The filter handler is invoked by SLAAC module when it is about to add a SLAAC address based on a prefix. Its boolean return value determines whether the address is filtered (not added) or not.

Definition at line `790` of file `include/openthread/ip6.h`

otIp6RegisterMulticastListenersCallback

```
typedef void(* otIp6RegisterMulticastListenersCallback) (void *aContext, otError aError, uint8_t aMlrStatus, const otIp6Address *aFailedAddresses, uint8_t aFailedAddressNum) )(void *aContext, otError aError, uint8_t aMlrStatus, const otIp6Address *aFailedAddresses, uint8_t aFailedAddressNum)
```

Pointer is called with results of `otIp6RegisterMulticastListeners`.

Parameters

[in]	aContext	A pointer to the user context.
[in]	aError	OT_ERROR_NONE when successfully sent MLR.req and received MLR.rsp, OT_ERROR_RESPONSE_TIMEOUT when failed to receive MLR.rsp, OT_ERROR_PARSE when failed to parse MLR.rsp.
[in]	aMlrStatus	The Multicast Listener Registration status when `aError` is OT_ERROR_NONE.
[in]	aFailedAddresses	A pointer to the failed IPv6 addresses when `aError` is OT_ERROR_NONE.
[in]	aFailedAddressNum	The number of failed IPv6 addresses when `aError` is OT_ERROR_NONE.

See Also

- otIp6RegisterMulticastListeners

Definition at line `824` of file `include/openthread/ip6.h`

otPacketsAndBytes

```
typedef struct otPacketsAndBytes otPacketsAndBytes
```

Represents the counters for packets and bytes.

Definition at line `897` of file `include/openthread/ip6.h`

otBorderRoutingCounters

```
typedef struct otBorderRoutingCounters otBorderRoutingCounters
```

Represents the counters of packets forwarded via Border Routing.

Definition at line 915 of file include/openthread/ip6.h

Variable Documentation

OT_TOOL_PACKED_END

```
OT_TOOL_PACKED_BEGIN struct otIp6Prefix OT_TOOL_PACKED_END
```

Definition at line 77 of file include/openthread/ip6.h

Function Documentation

otIp6SetEnabled

```
otError otIp6SetEnabled (otInstance *aInstance, bool aEnabled)
```

Brings the IPv6 interface up or down.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	TRUE to enable IPv6, FALSE otherwise.

Call this to enable or disable IPv6 communication.

Definition at line 277 of file include/openthread/ip6.h

otIp6IsEnabled

```
bool otIp6IsEnabled (otInstance *aInstance)
```

Indicates whether or not the IPv6 interface is up.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 288 of file include/openthread/ip6.h

otIp6AddUnicastAddress

```
otError otIp6AddUnicastAddress (otInstance *aInstance, const otNetifAddress *aAddress)
```

Adds a Network Interface Address to the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[in]	aAddress	A pointer to a Network Interface Address.

The passed-in instance `aAddress` is copied by the Thread interface. The Thread interface only supports a fixed number of externally added unicast addresses. See `OPENTHREAD_CONFIG_IP6_MAX_EXT_UCAST_ADDRS` .

Definition at line `304` of file `include/openthread/ip6.h`

otIp6RemoveUnicastAddress

> otError otIp6RemoveUnicastAddress (otInstance *aInstance, const otIp6Address *aAddress)

Removes a Network Interface Address from the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aAddress	A pointer to an IP Address.

Definition at line `317` of file `include/openthread/ip6.h`

otIp6GetUnicastAddresses

> const otNetifAddress * otIp6GetUnicastAddresses (otInstance *aInstance)

Gets the list of IPv6 addresses assigned to the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the first Network Interface Address.

Definition at line `327` of file `include/openthread/ip6.h`

otIp6HasUnicastAddress

> bool otIp6HasUnicastAddress (otInstance *aInstance, const otIp6Address *aAddress)

Indicates whether or not a unicast IPv6 address is assigned to the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aAddress	A pointer to the unicast address.

Definition at line `339` of file `include/openthread/ip6.h`

otIp6SubscribeMulticastAddress

> otError otIp6SubscribeMulticastAddress (otInstance *aInstance, const otIp6Address *aAddress)

Subscribes the Thread interface to a Network Interface Multicast Address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aAddress	A pointer to an IP Address.

The passed in instance `aAddress` will be copied by the Thread interface. The Thread interface only supports a fixed number of externally added multicast addresses. See `OPENTHREAD_CONFIG_IP6_MAX_EXT_MCAST_ADDRS`.

Definition at line `358` of file `include/openthread/ip6.h`

otIp6UnsubscribeMulticastAddress

> otError otIp6UnsubscribeMulticastAddress (otInstance *aInstance, const otIp6Address *aAddress)

Unsubscribes the Thread interface to a Network Interface Multicast Address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aAddress	A pointer to an IP Address.

Definition at line `371` of file `include/openthread/ip6.h`

otIp6GetMulticastAddresses

> const otNetifMulticastAddress * otIp6GetMulticastAddresses (otInstance *aInstance)

Gets the list of IPv6 multicast addresses subscribed to the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the first Network Interface Multicast Address.

Definition at line `381` of file `include/openthread/ip6.h`

otIp6IsMulticastPromiscuousEnabled

> bool otIp6IsMulticastPromiscuousEnabled (otInstance *aInstance)

Checks if multicast promiscuous mode is enabled on the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

See Also

- otIp6SetMulticastPromiscuousEnabled

Definition at line `391` of file `include/openthread/ip6.h`

otIp6SetMulticastPromiscuousEnabled

> void otIp6SetMulticastPromiscuousEnabled (otInstance *aInstance, bool aEnabled)

Enables or disables multicast promiscuous mode on the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	TRUE to enable Multicast Promiscuous mode, FALSE otherwise.

See Also

- otIp6IsMulticastPromiscuousEnabled

Definition at line `402` of file `include/openthread/ip6.h`

otIp6NewMessage

> otMessage * otIp6NewMessage (otInstance *aInstance, const otMessageSettings *aSettings)

Allocate a new message buffer for sending an IPv6 message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSettings	A pointer to the message settings or NULL to set default settings.

Note

- If `aSettings` is 'NULL', the link layer security is enabled and the message priority is set to OT_MESSAGE_PRIORITY_NORMAL by default.

Returns

- A pointer to the message buffer or NULL if no message buffers are available or parameters are invalid.

See Also

- otMessageFree

Definition at line `418` of file `include/openthread/ip6.h`

otIp6NewMessageFromBuffer

> otMessage * otIp6NewMessageFromBuffer (otInstance *aInstance, const uint8_t *aData, uint16_t aDataLength, const otMessageSettings *aSettings)

Allocate a new message buffer and write the IPv6 datagram to the message buffer for sending an IPv6 message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aData	A pointer to the IPv6 datagram buffer.
[in]	aDataLength	The size of the IPv6 datagram buffer pointed by `aData` .
[in]	aSettings	A pointer to the message settings or NULL to set default settings.

Note

If `aSettings` is NULL, the link layer security is enabled and the message priority is obtained from IPv6 message itself. If `aSettings` is not NULL, the `aSetting->mPriority` is ignored and obtained from IPv6 message itself.

Returns

- A pointer to the message or NULL if malformed IPv6 header or insufficient message buffers are available.

See Also

- otMessageFree

Definition at line `437` of file `include/openthread/ip6.h`

otIp6SetReceiveCallback

```
void otIp6SetReceiveCallback (otInstance *aInstance, otIp6ReceiveCallback aCallback, void *aCallbackContext)
```

Registers a callback to provide received IPv6 datagrams.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function that is called when an IPv6 datagram is received or NULL to disable the callback.
[in]	aCallbackContext	A pointer to application-specific context.

By default, this callback does not pass Thread control traffic. See otIp6SetReceiveFilterEnabled() to change the Thread control traffic filter setting.

See Also

- otIp6IsReceiveFilterEnabled
- otIp6SetReceiveFilterEnabled

Definition at line `468` of file `include/openthread/ip6.h`

otIp6SetAddressCallback

```
void otIp6SetAddressCallback (otInstance *aInstance, otIp6AddressCallback aCallback, void *aCallbackContext)
```

Registers a callback to notify internal IPv6 address changes.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function that is called when an internal IPv6 address is added or removed. NULL to disable the callback.
[in]	aCallbackContext	A pointer to application-specific context.

Definition at line `501` of file `include/openthread/ip6.h`

otIp6IsReceiveFilterEnabled

```
bool otIp6IsReceiveFilterEnabled (otInstance *aInstance)
```

Indicates whether or not Thread control traffic is filtered out when delivering IPv6 datagrams via the callback specified in otIp6SetReceiveCallback().

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- TRUE if Thread control traffic is filtered out, FALSE otherwise.

See Also

- otIp6SetReceiveCallback
- otIp6SetReceiveFilterEnabled

Definition at line 515 of file include/openthread/ip6.h

otIp6SetReceiveFilterEnabled

```
void otIp6SetReceiveFilterEnabled (otInstance *aInstance, bool aEnabled)
```

Sets whether or not Thread control traffic is filtered out when delivering IPv6 datagrams via the callback specified in otIp6SetReceiveCallback().

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	TRUE if Thread control traffic is filtered out, FALSE otherwise.

See Also

- otIp6SetReceiveCallback
- otIsReceiveIp6FilterEnabled

Definition at line 528 of file include/openthread/ip6.h

otIp6Send

```
otError otIp6Send (otInstance *aInstance, otMessage *aMessage)
```

Sends an IPv6 datagram via the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the message buffer containing the IPv6 datagram.

The caller transfers ownership of aMessage when making this call. OpenThread will free aMessage when processing is complete, including when a value other than OT_ERROR_NONE is returned.

Definition at line 550 of file include/openthread/ip6.h

otIp6AddUnsecurePort

```
otError otIp6AddUnsecurePort (otInstance *aInstance, uint16_t aPort)
```

Adds a port to the allowed unsecured port list.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPort	The port value.

Definition at line `563` of file `include/openthread/ip6.h`

otIp6RemoveUnsecurePort

```
otError otIp6RemoveUnsecurePort (otInstance *aInstance, uint16_t aPort)
```

Removes a port from the allowed unsecure port list.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPort	The port value.

Note

- This function removes `aPort` by overwriting `aPort` with the element after `aPort` in the internal port list. Be careful when calling otIp6GetUnsecurePorts() followed by otIp6RemoveUnsecurePort() to remove unsecure ports.

Definition at line `580` of file `include/openthread/ip6.h`

otIp6RemoveAllUnsecurePorts

```
void otIp6RemoveAllUnsecurePorts (otInstance *aInstance)
```

Removes all ports from the allowed unsecure port list.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `588` of file `include/openthread/ip6.h`

otIp6GetUnsecurePorts

```
const uint16_t * otIp6GetUnsecurePorts (otInstance *aInstance, uint8_t *aNumEntries)
```

Returns a pointer to the unsecure port list.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aNumEntries	The number of entries in the list.

Note

- Port value 0 is used to indicate an invalid entry.

Returns

- A pointer to the unsecure port list.

Definition at line 601 of file include/openthread/ip6.h

otIp6IsAddressEqual

> bool otIp6IsAddressEqual (const otIp6Address *aFirst, const otIp6Address *aSecond)

Test if two IPv6 addresses are the same.

Parameters

[in]	aFirst	A pointer to the first IPv6 address to compare.
[in]	aSecond	A pointer to the second IPv6 address to compare.

Definition at line 613 of file include/openthread/ip6.h

otIp6ArePrefixesEqual

> bool otIp6ArePrefixesEqual (const otIp6Prefix *aFirst, const otIp6Prefix *aSecond)

Test if two IPv6 prefixes are the same.

Parameters

[in]	aFirst	A pointer to the first IPv6 prefix to compare.
[in]	aSecond	A pointer to the second IPv6 prefix to compare.

Definition at line 625 of file include/openthread/ip6.h

otIp6AddressFromString

> otError otIp6AddressFromString (const char *aString, otIp6Address *aAddress)

Converts a human-readable IPv6 address string into a binary representation.

Parameters

[in]	aString	A pointer to a NULL-terminated string.
[out]	aAddress	A pointer to an IPv6 address.

Definition at line 637 of file include/openthread/ip6.h

otIp6PrefixFromString

> otError otIp6PrefixFromString (const char *aString, otIp6Prefix *aPrefix)

Converts a human-readable IPv6 prefix string into a binary representation.

Parameters

[in]	aString	A pointer to a NULL-terminated string.
[out]	aPrefix	A pointer to an IPv6 prefix.

The `aString` parameter should be a string in the format "<address>/<plen>", where `<address>` is an IPv6 address and `<plen>` is a prefix length.

Definition at line 652 of file include/openthread/ip6.h

otIp6AddressToString

> void otIp6AddressToString (const otIp6Address *aAddress, char *aBuffer, uint16_t aSize)

Converts a given IPv6 address to a human-readable string.

Parameters

[in]	aAddress	A pointer to an IPv6 address (MUST NOT be NULL).
[out]	aBuffer	A pointer to a char array to output the string (MUST NOT be NULL).
[in]	aSize	The size of aBuffer (in bytes). Recommended to use OT_IP6_ADDRESS_STRING_SIZE.

The IPv6 address string is formatted as 16 hex values separated by ':' (i.e., "%x:%x:%x:...:%x").

If the resulting string does not fit in aBuffer (within its aSize characters), the string will be truncated but the outputted string is always null-terminated.

Definition at line 669 of file include/openthread/ip6.h

otIp6SockAddrToString

> void otIp6SockAddrToString (const otSockAddr *aSockAddr, char *aBuffer, uint16_t aSize)

Converts a given IPv6 socket address to a human-readable string.

Parameters

[in]	aSockAddr	A pointer to an IPv6 socket address (MUST NOT be NULL).
[out]	aBuffer	A pointer to a char array to output the string (MUST NOT be NULL).
[in]	aSize	The size of aBuffer (in bytes). Recommended to use OT_IP6_SOCK_ADDR_STRING_SIZE.

The IPv6 socket address string is formatted as [address]: port where address is shown as 16 hex values separated by : and port is the port number in decimal format, for example "[%x:%x:...:%x]:%u".

If the resulting string does not fit in aBuffer (within its aSize characters), the string will be truncated but the outputted string is always null-terminated.

Definition at line 688 of file include/openthread/ip6.h

otIp6PrefixToString

> void otIp6PrefixToString (const otIp6Prefix *aPrefix, char *aBuffer, uint16_t aSize)

Converts a given IPv6 prefix to a human-readable string.

Parameters

[in]	aPrefix	A pointer to an IPv6 prefix (MUST NOT be NULL).
[out]	aBuffer	A pointer to a char array to output the string (MUST NOT be NULL).
[in]	aSize	The size of aBuffer (in bytes). Recommended to use OT_IP6_PREFIX_STRING_SIZE.

The IPv6 address string is formatted as "%x:%x:%x:...[::]/plen".

If the resulting string does not fit in `aBuffer` (within its `aSize` characters), the string will be truncated but the outputted string is always null-terminated.

Definition at line `705` of file `include/openthread/ip6.h`

otIp6PrefixMatch

```
uint8_t otIp6PrefixMatch (const otIp6Address *aFirst, const otIp6Address *aSecond)
```

Returns the prefix match length (bits) for two IPv6 addresses.

Parameters

[in]	aFirst	A pointer to the first IPv6 address.
[in]	aSecond	A pointer to the second IPv6 address.

Returns

- The prefix match length in bits.

Definition at line `716` of file `include/openthread/ip6.h`

otIp6GetPrefix

```
void otIp6GetPrefix (const otIp6Address *aAddress, uint8_t aLength, otIp6Prefix *aPrefix)
```

Gets a prefix with `aLength` from `aAddress` .

Parameters

[in]	aAddress	A pointer to an IPv6 address.
[in]	aLength	The length of prefix in bits.
[out]	aPrefix	A pointer to output the IPv6 prefix.

Definition at line `726` of file `include/openthread/ip6.h`

otIp6IsAddressUnspecified

```
bool otIp6IsAddressUnspecified (const otIp6Address *aAddress)
```

Indicates whether or not a given IPv6 address is the Unspecified Address.

Parameters

[in]	aAddress	A pointer to an IPv6 address.

Definition at line `737` of file `include/openthread/ip6.h`

otIp6SelectSourceAddress

```
otError otIp6SelectSourceAddress (otInstance *aInstance, otMessageInfo *aMessageInfo)
```

Perform OpenThread source address selection.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aMessageInfo	A pointer to the message information.

Definition at line `749` of file `include/openthread/ip6.h`

otIp6IsSlaacEnabled

```
bool otIp6IsSlaacEnabled (otInstance *aInstance)
```

Indicates whether the SLAAC module is enabled or not.

Parameters

N/A	aInstance	

`OPENTHREAD_CONFIG_IP6_SLAAC_ENABLE` build-time feature must be enabled.

Definition at line `760` of file `include/openthread/ip6.h`

otIp6SetSlaacEnabled

```
void otIp6SetSlaacEnabled (otInstance *aInstance, bool aEnabled)
```

Enables/disables the SLAAC module.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	TRUE to enable, FALSE to disable.

`OPENTHREAD_CONFIG_IP6_SLAAC_ENABLE` build-time feature must be enabled.

When SLAAC module is enabled, SLAAC addresses (based on on-mesh prefixes in Network Data) are added to the interface. When SLAAC module is disabled any previously added SLAAC address is removed.

Definition at line `774` of file `include/openthread/ip6.h`

otIp6SetSlaacPrefixFilter

```
void otIp6SetSlaacPrefixFilter (otInstance *aInstance, otIp6SlaacPrefixFilter aFilter)
```

Sets the SLAAC module filter handler.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aFilter	A pointer to SLAAC prefix filter handler, or NULL to disable filtering.

`OPENTHREAD_CONFIG_IP6_SLAAC_ENABLE` build-time feature must be enabled.

The filter handler is called by SLAAC module when it is about to add a SLAAC address based on a prefix to decide whether the address should be added or not.

A NULL filter handler disables filtering and allows all SLAAC addresses to be added.

If this function is not called, the default filter used by SLAAC module will be NULL (filtering is disabled).

Definition at line `808` of file `include/openthread/ip6.h`

otIp6RegisterMulticastListeners

> otError otIp6RegisterMulticastListeners (otInstance *aInstance, const otIp6Address *aAddresses, uint8_t aAddressNum, const uint32_t *aTimeout, otIp6RegisterMulticastListenersCallback aCallback, void *aContext)

Registers Multicast Listeners to Primary Backbone Router.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aAddresses	A Multicast Address Array to register.
[in]	aAddressNum	The number of Multicast Address to register (0 if `aAddresses` is NULL).
[in]	aTimeout	A pointer to the timeout value (in seconds) to be included in MLR.req. A timeout value of 0 removes the corresponding Multicast Listener. If NULL, MLR.req would have no Timeout Tlv by default.
[in]	aCallback	A pointer to the callback function.
[in]	aContext	A pointer to the user context.

`OPENTHREAD_CONFIG_TMF_PROXY_MLR_ENABLE` and `OPENTHREAD_CONFIG_COMMISSIONER_ENABLE` must be enabled.

See Also

- otIp6RegisterMulticastListenersCallback

Definition at line `858` of file `include/openthread/ip6.h`

otIp6SetMeshLocalIid

> otError otIp6SetMeshLocalIid (otInstance *aInstance, const otIp6InterfaceIdentifier *aIid)

Sets the Mesh Local IID (for test purpose).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aIid	A pointer to the Mesh Local IID to set.

Requires `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE`.

Definition at line `877` of file `include/openthread/ip6.h`

otIp6ProtoToString

> const char * otIp6ProtoToString (uint8_t aIpProto)

Converts a given IP protocol number to a human-readable string.

Parameters

[in]	aIpProto	An IP protocol number (`OT_IP6_PROTO_*` enumeration).

Returns

- A string representing `aIpProto` .

Definition at line `887` of file `include/openthread/ip6.h`

otIp6GetBorderRoutingCounters

```
const otBorderRoutingCounters * otIp6GetBorderRoutingCounters (otInstance *aInstance)
```

Gets the Border Routing counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

`OPENTHREAD_CONFIG_IP6_BR_COUNTERS_ENABLE` build-time feature must be enabled.

Returns

- A pointer to the Border Routing counters.

Definition at line `927` of file `include/openthread/ip6.h`

otIp6ResetBorderRoutingCounters

```
void otIp6ResetBorderRoutingCounters (otInstance *aInstance)
```

Resets the Border Routing counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `935` of file `include/openthread/ip6.h`

Macro Definition Documentation

OT_IP6_PREFIX_SIZE

```
#define OT_IP6_PREFIX_SIZE
```

Value:

```
8
```

Size of an IPv6 prefix (bytes)

Definition at line `55` of file `include/openthread/ip6.h`

OT_IP6_PREFIX_BITSIZE

```
#define OT_IP6_PREFIX_BITSIZE
```

Value:

```
(OT_IP6_PREFIX_SIZE * 8)
```

Size of an IPv6 prefix (bits)

Definition at line 56 of file include/openthread/ip6.h

OT_IP6_IID_SIZE

```
#define OT_IP6_IID_SIZE
```

Value:

```
8
```

Size of an IPv6 Interface Identifier (bytes)

Definition at line 57 of file include/openthread/ip6.h

OT_IP6_ADDRESS_SIZE

```
#define OT_IP6_ADDRESS_SIZE
```

Value:

```
16
```

Size of an IPv6 address (bytes)

Definition at line 58 of file include/openthread/ip6.h

OT_IP6_HEADER_SIZE

```
#define OT_IP6_HEADER_SIZE
```

Value:

```
40
```

Size of an IPv6 header (bytes)

Definition at line 59 of file include/openthread/ip6.h

OT_IP6_HEADER_PROTO_OFFSET

```
#define OT_IP6_HEADER_PROTO_OFFSET
```

Value:

```
6
```

Offset of the proto field in the IPv6 header (bytes)

Definition at line 60 of file include/openthread/ip6.h

OT_IP6_ADDRESS_STRING_SIZE

```
#define OT_IP6_ADDRESS_STRING_SIZE
```

Value:

```
40
```

Recommended size for string representation of an IPv6 address.

Definition at line `654` of file `include/openthread/ip6.h`

OT_IP6_SOCK_ADDR_STRING_SIZE

```
#define OT_IP6_SOCK_ADDR_STRING_SIZE
```

Value:

```
48
```

Recommended size for string representation of an IPv6 socket address.

Definition at line `671` of file `include/openthread/ip6.h`

OT_IP6_PREFIX_STRING_SIZE

```
#define OT_IP6_PREFIX_STRING_SIZE
```

Value:

```
45
```

Recommended size for string representation of an IPv6 prefix.

Definition at line `690` of file `include/openthread/ip6.h`

OT_IP6_MAX_MLR_ADDRESSES

```
#define OT_IP6_MAX_MLR_ADDRESSES
```

Value:

```
15
```

Max number of IPv6 addresses supported by Multicast Listener Registration.

Definition at line `830` of file `include/openthread/ip6.h`

otIp6InterfaceIdentifier

Represents the Interface Identifier of an IPv6 address.

Modules

otIp6InterfaceIdentifier::OT_TOOL_PACKED_FIELD

Public Attributes

union otIp6InterfaceIdentifier::OT_TOOL_PACKED_FIELD	mFields The Interface Identifier accessor fields.

Public Attribute Documentation

mFields

union otIp6InterfaceIdentifier::OT_TOOL_PACKED_FIELD otIp6InterfaceIdentifier::mFields

The Interface Identifier accessor fields.

Definition at line 76 of file include/openthread/ip6.h

otIp6InterfaceIdentifier

Public Attributes

uint8_t	m8 8-bit fields
uint16_t	m16 16-bit fields
uint32_t	m32 32-bit fields

Public Attribute Documentation

m8

```
uint8_t otIp6InterfaceIdentifier::OT_TOOL_PACKED_FIELD::m8[OT_IP6_IID_SIZE]
```

8-bit fields

Definition at line `73` of file `include/openthread/ip6.h`

m16

```
uint16_t otIp6InterfaceIdentifier::OT_TOOL_PACKED_FIELD::m16[OT_IP6_IID_SIZE/sizeof(uint16_t)]
```

16-bit fields

Definition at line `74` of file `include/openthread/ip6.h`

m32

```
uint32_t otIp6InterfaceIdentifier::OT_TOOL_PACKED_FIELD::m32[OT_IP6_IID_SIZE/sizeof(uint32_t)]
```

32-bit fields

Definition at line `75` of file `include/openthread/ip6.h`

otIp6NetworkPrefix

Represents the Network Prefix of an IPv6 address (most significant 64 bits of the address).

Public Attributes

uint8_t	m8 The Network Prefix.

Public Attribute Documentation

m8

```
uint8_t otIp6NetworkPrefix::m8[OT_IP6_PREFIX_SIZE]
```

The Network Prefix.

Definition at line 94 of file include/openthread/ip6.h

otIp6AddressComponents

Represents the components of an IPv6 address.

Public Attributes

otIp6NetworkPrefix	mNetworkPrefix The Network Prefix (most significant 64 bits of the address)
otIp6InterfaceIdentifier	mlid The Interface Identifier (least significant 64 bits of the address)

Public Attribute Documentation

mNetworkPrefix

```
otIp6NetworkPrefix otIp6AddressComponents::mNetworkPrefix
```

The Network Prefix (most significant 64 bits of the address)

Definition at line 112 of file include/openthread/ip6.h

mlid

```
otIp6InterfaceIdentifier otIp6AddressComponents::mlid
```

The Interface Identifier (least significant 64 bits of the address)

Definition at line 113 of file include/openthread/ip6.h

otIp6Address

Represents an IPv6 address.

Modules

otIp6Address::OT_TOOL_PACKED_FIELD

Public Attributes

union otIp6Address::OT_TOOL_PACKED_FIELD	mFields IPv6 accessor fields.

Public Attribute Documentation

mFields

```
union otIp6Address::OT_TOOL_PACKED_FIELD otIp6Address::mFields
```

IPv6 accessor fields.

Definition at line 137 of file include/openthread/ip6.h

otIp6Address

Public Attributes

uint8_t	m8 8-bit fields
uint16_t	m16 16-bit fields
uint32_t	m32 32-bit fields
otIp6AddressComponents	mComponents IPv6 address components.

Public Attribute Documentation

m8

> uint8_t otIp6Address::OT_TOOL_PACKED_FIELD::m8[OT_IP6_ADDRESS_SIZE]

8-bit fields

Definition at line 133 of file include/openthread/ip6.h

m16

> uint16_t otIp6Address::OT_TOOL_PACKED_FIELD::m16[OT_IP6_ADDRESS_SIZE/sizeof(uint16_t)]

16-bit fields

Definition at line 134 of file include/openthread/ip6.h

m32

> uint32_t otIp6Address::OT_TOOL_PACKED_FIELD::m32[OT_IP6_ADDRESS_SIZE/sizeof(uint32_t)]

32-bit fields

Definition at line 135 of file include/openthread/ip6.h

mComponents

> otIp6AddressComponents otIp6Address::OT_TOOL_PACKED_FIELD::mComponents

IPv6 address components.

Definition at line 136 of file include/openthread/ip6.h

otIp6Prefix

Represents an IPv6 prefix.

Public Attributes

otIp6Address	mPrefix The IPv6 prefix.
uint8_t	mLength The IPv6 prefix length (in bits).

Public Attribute Documentation

mPrefix

```
otIp6Address otIp6Prefix::mPrefix
```

The IPv6 prefix.

Definition at line 155 of file include/openthread/ip6.h

mLength

```
uint8_t otIp6Prefix::mLength
```

The IPv6 prefix length (in bits).

Definition at line 156 of file include/openthread/ip6.h

otNetifAddress

Represents an IPv6 network interface unicast address.

Public Attributes

otIp6Address	mAddress The IPv6 unicast address.
uint8_t	mPrefixLength The Prefix length (in bits).
uint8_t	mAddressOrigin The IPv6 address origin.
bool	mPreferred TRUE if the address is preferred, FALSE otherwise.
bool	mValid TRUE if the address is valid, FALSE otherwise.
bool	mScopeOverrideValid TRUE if the mScopeOverride value is valid, FALSE otherwise.
unsigned int	mScopeOverride The IPv6 scope of this address.
bool	mRloc TRUE if the address is an RLOC, FALSE otherwise.
bool	mMeshLocal TRUE if the address is mesh-local, FALSE otherwise.
const struct otNetifAddress *	mNext A pointer to the next network interface address.

Public Attribute Documentation

mAddress

```
otIp6Address otNetifAddress::mAddress
```

The IPv6 unicast address.

Definition at line `183` of file `include/openthread/ip6.h`

mPrefixLength

```
uint8_t otNetifAddress::mPrefixLength
```

The Prefix length (in bits).

Definition at line `184` of file `include/openthread/ip6.h`

mAddressOrigin

```
uint8_t otNetifAddress::mAddressOrigin
```

The IPv6 address origin.

Definition at line 185 of file include/openthread/ip6.h

mPreferred

```
bool otNetifAddress::mPreferred
```

TRUE if the address is preferred, FALSE otherwise.

Definition at line 186 of file include/openthread/ip6.h

mValid

```
bool otNetifAddress::mValid
```

TRUE if the address is valid, FALSE otherwise.

Definition at line 187 of file include/openthread/ip6.h

mScopeOverrideValid

```
bool otNetifAddress::mScopeOverrideValid
```

TRUE if the mScopeOverride value is valid, FALSE otherwise.

Definition at line 188 of file include/openthread/ip6.h

mScopeOverride

```
unsigned int otNetifAddress::mScopeOverride
```

The IPv6 scope of this address.

Definition at line 189 of file include/openthread/ip6.h

mRloc

```
bool otNetifAddress::mRloc
```

TRUE if the address is an RLOC, FALSE otherwise.

Definition at line 190 of file include/openthread/ip6.h

mMeshLocal

```
bool otNetifAddress::mMeshLocal
```

TRUE if the address is mesh-local, FALSE otherwise.

Definition at line 191 of file include/openthread/ip6.h

mNext

```
const struct otNetifAddress* otNetifAddress::mNext
```

A pointer to the next network interface address.

Definition at line 192 of file include/openthread/ip6.h

otNetifMulticastAddress

Represents an IPv6 network interface multicast address.

Public Attributes

otIp6Address	mAddress The IPv6 multicast address.
const struct otNetifMulticastAddress *	mNext A pointer to the next network interface multicast address.

Public Attribute Documentation

mAddress

```
otIp6Address otNetifMulticastAddress::mAddress
```

The IPv6 multicast address.

Definition at line 201 of file include/openthread/ip6.h

mNext

```
const struct otNetifMulticastAddress* otNetifMulticastAddress::mNext
```

A pointer to the next network interface multicast address.

Definition at line 202 of file include/openthread/ip6.h

otSockAddr

Represents an IPv6 socket address.

Public Attributes

otIp6Address	mAddress An IPv6 address.
uint16_t	mPort A transport-layer port.

Public Attribute Documentation

mAddress

```
otIp6Address otSockAddr::mAddress
```

An IPv6 address.

Definition at line 211 of file include/openthread/ip6.h

mPort

```
uint16_t otSockAddr::mPort
```

A transport-layer port.

Definition at line 212 of file include/openthread/ip6.h

otMessageInfo

Represents the local and peer IPv6 socket addresses.

Public Attributes

otIp6Address	mSockAddr The local IPv6 address.
otIp6Address	mPeerAddr The peer IPv6 address.
uint16_t	mSockPort The local transport-layer port.
uint16_t	mPeerPort The peer transport-layer port.
const void *	mLinkInfo A pointer to link-specific information.
uint8_t	mHopLimit The IPv6 Hop Limit value.
uint8_t	mEcn The ECN status of the packet, represented as in the IPv6 header.
bool	mIsHostInterface TRUE if packets sent/received via host interface, FALSE otherwise.
bool	mAllowZeroHopLimit TRUE to allow IPv6 Hop Limit 0 in `mHopLimit`, FALSE otherwise.
bool	mMulticastLoop TRUE to allow looping back multicast, FALSE otherwise.

Public Attribute Documentation

mSockAddr

```
otIp6Address otMessageInfo::mSockAddr
```

The local IPv6 address.

Definition at line `233` of file `include/openthread/ip6.h`

mPeerAddr

```
otIp6Address otMessageInfo::mPeerAddr
```

The peer IPv6 address.

Definition at line `234` of file `include/openthread/ip6.h`

mSockPort

```
uint16_t otMessageInfo::mSockPort
```

The local transport-layer port.

Definition at line `235` of file `include/openthread/ip6.h`

mPeerPort

```
uint16_t otMessageInfo::mPeerPort
```

The peer transport-layer port.

Definition at line `236` of file `include/openthread/ip6.h`

mLinkInfo

```
const void* otMessageInfo::mLinkInfo
```

A pointer to link-specific information.

Definition at line `237` of file `include/openthread/ip6.h`

mHopLimit

```
uint8_t otMessageInfo::mHopLimit
```

The IPv6 Hop Limit value.

Only applies if `mAllowZeroHopLimit` is FALSE. If `0`, IPv6 Hop Limit is default value `OPENTHREAD_CONFIG_IP6_HOP_LIMIT_DEFAULT`. Otherwise, specifies the IPv6 Hop Limit.

Definition at line `238` of file `include/openthread/ip6.h`

mEcn

```
uint8_t otMessageInfo::mEcn
```

The ECN status of the packet, represented as in the IPv6 header.

Definition at line `241` of file `include/openthread/ip6.h`

mIsHostInterface

```
bool otMessageInfo::mIsHostInterface
```

TRUE if packets sent/received via host interface, FALSE otherwise.

Definition at line `242` of file `include/openthread/ip6.h`

mAllowZeroHopLimit

```
bool otMessageInfo::mAllowZeroHopLimit
```

TRUE to allow IPv6 Hop Limit 0 in `mHopLimit`, FALSE otherwise.

Definition at line `243` of file `include/openthread/ip6.h`

mMulticastLoop

```
bool otMessageInfo::mMulticastLoop
```

TRUE to allow looping back multicast, FALSE otherwise.

Definition at line `244` of file `include/openthread/ip6.h`

otIp6AddressInfo

Represents IPv6 address information.

Public Attributes

const otIp6Address *	mAddress A pointer to the IPv6 address.
uint8_t	mPrefixLength The prefix length of mAddress if it is a unicast address.
uint8_t	mScope The scope of this address.
bool	mPreferred Whether this is a preferred address.

Public Attribute Documentation

mAddress

```
const otIp6Address* otIp6AddressInfo::mAddress
```

A pointer to the IPv6 address.

Definition at line 476 of file include/openthread/ip6.h

mPrefixLength

```
uint8_t otIp6AddressInfo::mPrefixLength
```

The prefix length of mAddress if it is a unicast address.

Definition at line 477 of file include/openthread/ip6.h

mScope

```
uint8_t otIp6AddressInfo::mScope
```

The scope of this address.

Definition at line 478 of file include/openthread/ip6.h

mPreferred

```
bool otIp6AddressInfo::mPreferred
```

Whether this is a preferred address.

Definition at line 479 of file include/openthread/ip6.h

otPacketsAndBytes

Represents the counters for packets and bytes.

Public Attributes

uint64_t	mPackets The number of packets.
uint64_t	mBytes The number of bytes.

Public Attribute Documentation

mPackets

```
uint64_t otPacketsAndBytes::mPackets
```

The number of packets.

Definition at line `895` of file `include/openthread/ip6.h`

mBytes

```
uint64_t otPacketsAndBytes::mBytes
```

The number of bytes.

Definition at line `896` of file `include/openthread/ip6.h`

otBorderRoutingCounters

Represents the counters of packets forwarded via Border Routing.

Public Attributes

otPacketsAndBytes	mInboundUnicast The counters for inbound unicast.
otPacketsAndBytes	mInboundMulticast The counters for inbound multicast.
otPacketsAndBytes	mOutboundUnicast The counters for outbound unicast.
otPacketsAndBytes	mOutboundMulticast The counters for outbound multicast.
uint32_t	mRaRx The number of received RA packets.
uint32_t	mRaTxSuccess The number of RA packets successfully transmitted.
uint32_t	mRaTxFailure The number of RA packets failed to transmit.
uint32_t	mRsRx The number of received RS packets.
uint32_t	mRsTxSuccess The number of RS packets successfully transmitted.
uint32_t	mRsTxFailure The number of RS packets failed to transmit.

Public Attribute Documentation

mInboundUnicast

```
otPacketsAndBytes otBorderRoutingCounters::mInboundUnicast
```

The counters for inbound unicast.

Definition at line `905` of file `include/openthread/ip6.h`

mInboundMulticast

```
otPacketsAndBytes otBorderRoutingCounters::mInboundMulticast
```

The counters for inbound multicast.

Definition at line `906` of file `include/openthread/ip6.h`

mOutboundUnicast

```
otPacketsAndBytes otBorderRoutingCounters::mOutboundUnicast
```

The counters for outbound unicast.

Definition at line `907` of file `include/openthread/ip6.h`

mOutboundMulticast

```
otPacketsAndBytes otBorderRoutingCounters::mOutboundMulticast
```

The counters for outbound multicast.

Definition at line `908` of file `include/openthread/ip6.h`

mRaRx

```
uint32_t otBorderRoutingCounters::mRaRx
```

The number of received RA packets.

Definition at line `909` of file `include/openthread/ip6.h`

mRaTxSuccess

```
uint32_t otBorderRoutingCounters::mRaTxSuccess
```

The number of RA packets successfully transmitted.

Definition at line `910` of file `include/openthread/ip6.h`

mRaTxFailure

```
uint32_t otBorderRoutingCounters::mRaTxFailure
```

The number of RA packets failed to transmit.

Definition at line `911` of file `include/openthread/ip6.h`

mRsRx

```
uint32_t otBorderRoutingCounters::mRsRx
```

The number of received RS packets.

Definition at line `912` of file `include/openthread/ip6.h`

mRsTxSuccess

uint32_t otBorderRoutingCounters::mRsTxSuccess

The number of RS packets successfully transmitted.

Definition at line 913 of file include/openthread/ip6.h

mRsTxFailure

uint32_t otBorderRoutingCounters::mRsTxFailure

The number of RS packets failed to transmit.

Definition at line 914 of file include/openthread/ip6.h

NAT64

NAT64

This module includes functions and structs for the NAT64 function on the border router.

These functions are only available when `OPENTHREAD_CONFIG_NAT64_BORDER_ROUTING_ENABLE` is enabled.

Modules

otIp4Address

otIp4Cidr

otNat64Counters

otNat64ProtocolCounters

otNat64ErrorCounters

otNat64AddressMapping

otNat64AddressMappingIterator

Enumerations

enum otNat64DropReason {

OT_NAT64_DROP_REASON_UNKNOWN = 0
OT_NAT64_DROP_REASON_ILLEGAL_PACKET
OT_NAT64_DROP_REASON_UNSUPPORTED_PROTO
OT_NAT64_DROP_REASON_NO_MAPPING
OT_NAT64_DROP_REASON_COUNT

}

Packet drop reasons.

enum otNat64State {

OT_NAT64_STATE_DISABLED = 0
OT_NAT64_STATE_NOT_RUNNING
OT_NAT64_STATE_IDLE
OT_NAT64_STATE_ACTIVE

}

States of NAT64.

Typedefs

typedef struct otIp4Address — otIp4Address

Represents an IPv4 address.

typedef struct otIp4Cidr — otIp4Cidr

typedef struct otNat64Counters — otNat64Counters

Represents the counters for NAT64.

typedef struct otNat64ProtocolCounters	otNat64ProtocolCounters Represents the counters for the protocols supported by NAT64.
typedef enum otNat64DropReason	otNat64DropReason Packet drop reasons.
typedef struct otNat64ErrorCounters	otNat64ErrorCounters Represents the counters of dropped packets due to errors when handling NAT64 packets.
typedef struct otNat64AddressMapping	otNat64AddressMapping Represents an address mapping record for NAT64.
typedef struct otNat64AddressMappingIterator	otNat64AddressMappingIterator Used to iterate through NAT64 address mappings.
typedef void(*	otNat64ReceiveIp4Callback)(otMessage *aMessage, void *aContext) Pointer is called when an IPv4 datagram (translated by NAT64 translator) is received.

Variables

OT_TOOL_PACKED_BEGIN struct otIp4Address	OT_TOOL_PACKED_END

Functions

void	otNat64GetCounters(otInstance *aInstance, otNat64ProtocolCounters *aCounters) Gets NAT64 translator counters.
void	otNat64GetErrorCounters(otInstance *aInstance, otNat64ErrorCounters *aCounters) Gets the NAT64 translator error counters.
void	otNat64InitAddressMappingIterator(otInstance *aInstance, otNat64AddressMappingIterator *aIterator) Initializes an `otNat64AddressMappingIterator`.
otError	otNat64GetNextAddressMapping(otInstance *aInstance, otNat64AddressMappingIterator *aIterator, otNat64AddressMapping *aMapping) Gets the next AddressMapping info (using an iterator).
otNat64State	otNat64GetTranslatorState(otInstance *aInstance) Gets the state of NAT64 translator.
otNat64State	otNat64GetPrefixManagerState(otInstance *aInstance) Gets the state of NAT64 prefix manager.
void	otNat64SetEnabled(otInstance *aInstance, bool aEnabled) Enable or disable NAT64 functions.
otMessage *	otIp4NewMessage(otInstance *aInstance, const otMessageSettings *aSettings) Allocate a new message buffer for sending an IPv4 message to the NAT64 translator.
otError	otNat64SetIp4Cidr(otInstance *aInstance, const otIp4Cidr *aCidr) Sets the CIDR used when setting the source address of the outgoing translated IPv4 packets.
otError	otNat64Send(otInstance *aInstance, otMessage *aMessage) Translates an IPv4 datagram to an IPv6 datagram and sends via the Thread interface.

void otNat64SetReceiveIp4Callback(otInstance *aInstance, otNat64ReceiveIp4Callback aCallback, void *aContext)
Registers a callback to provide received IPv4 datagrams.

otError otNat64GetCidr(otInstance *aInstance, otIp4Cidr *aCidr)
Gets the IPv4 CIDR configured in the NAT64 translator.

bool otIp4IsAddressEqual(const otIp4Address *aFirst, const otIp4Address *aSecond)
Test if two IPv4 addresses are the same.

void otIp4ExtractFromIp6Address(uint8_t aPrefixLength, const otIp6Address *aIp6Address, otIp4Address *aIp4Address)
Set `aIp4Address` by performing NAT64 address translation from `aIp6Address` as specified in RFC 6052.

void otIp4AddressToString(const otIp4Address *aAddress, char *aBuffer, uint16_t aSize)
Converts the address to a string.

otError otIp4CidrFromString(const char *aString, otIp4Cidr *aCidr)
Converts a human-readable IPv4 CIDR string into a binary representation.

void otIp4CidrToString(const otIp4Cidr *aCidr, char *aBuffer, uint16_t aSize)
Converts the IPv4 CIDR to a string.

otError otIp4AddressFromString(const char *aString, otIp4Address *aAddress)
Converts a human-readable IPv4 address string into a binary representation.

otError otNat64SynthesizeIp6Address(otInstance *aInstance, const otIp4Address *aIp4Address, otIp6Address *aIp6Address)
Sets the IPv6 address by performing NAT64 address translation from the preferred NAT64 prefix and the given IPv4 address as specified in RFC 6052.

Macros

#define OT_IP4_ADDRESS_SIZE 4
Size of an IPv4 address (bytes)

#define OT_IP4_ADDRESS_STRING_SIZE 17
Length of 000.000.000.000 plus a suffix NUL.

#define OT_IP4_CIDR_STRING_SIZE 20
Length of 000.000.000.000/00 plus a suffix NUL.

Enumeration Documentation

otNat64DropReason

> otNat64DropReason

Packet drop reasons.

Enumerator	
OT_NAT64_DROP_REASON_UNKNOWN	Packet drop for unknown reasons.
OT_NAT64_DROP_REASON_ILLEGAL_PACKET	Packet drop due to failed to parse the datagram.
OT_NAT64_DROP_REASON_UNSUPPORTED_PROTO	Packet drop due to unsupported IP protocol.
OT_NAT64_DROP_REASON_NO_MAPPING	Packet drop due to no mappings found or mapping pool exhausted.
OT_NAT64_DROP_REASON_COUNT	

Definition at line `119` of file `include/openthread/nat64.h`

otNat64State

```
otNat64State
```

States of NAT64.

Enumerator	
OT_NAT64_STATE_DISABLED	NAT64 is disabled.
OT_NAT64_STATE_NOT_RUNNING	NAT64 is enabled, but one or more dependencies of NAT64 are not running.
OT_NAT64_STATE_IDLE	NAT64 is enabled, but this BR is not an active NAT64 BR.
OT_NAT64_STATE_ACTIVE	The BR is publishing a NAT64 prefix and/or translating packets.

Definition at line 233 of file include/openthread/nat64.h

Typedef Documentation

otIp4Address

```
typedef struct otIp4Address otIp4Address
```

Represents an IPv4 address.

Definition at line 77 of file include/openthread/nat64.h

otIp4Cidr

```
typedef struct otIp4Cidr otIp4Cidr
```

Definition at line 89 of file include/openthread/nat64.h

otNat64Counters

```
typedef struct otNat64Counters otNat64Counters
```

Represents the counters for NAT64.

Definition at line 101 of file include/openthread/nat64.h

otNat64ProtocolCounters

```
typedef struct otNat64ProtocolCounters otNat64ProtocolCounters
```

Represents the counters for the protocols supported by NAT64.

Definition at line 113 of file include/openthread/nat64.h

otNat64DropReason

```
typedef enum otNat64DropReason otNat64DropReason
```

Packet drop reasons.

Definition at line 127 of file include/openthread/nat64.h

otNat64ErrorCounters

```
typedef struct otNat64ErrorCounters otNat64ErrorCounters
```

Represents the counters of dropped packets due to errors when handling NAT64 packets.

Definition at line 137 of file include/openthread/nat64.h

otNat64AddressMapping

```
typedef struct otNat64AddressMapping otNat64AddressMapping
```

Represents an address mapping record for NAT64.

Note

- The counters will be reset for each mapping session even for the same address pair. Applications can use `mId` to identify different sessions to calculate the packets correctly.

Definition at line 179 of file include/openthread/nat64.h

otNat64AddressMappingIterator

```
typedef struct otNat64AddressMappingIterator otNat64AddressMappingIterator
```

Used to iterate through NAT64 address mappings.

The fields in this type are opaque (intended for use by OpenThread core only) and therefore should not be accessed or used by caller.

Before using an iterator, it MUST be initialized using `otNat64AddressMappingIteratorInit()`.

Definition at line 193 of file include/openthread/nat64.h

otNat64ReceiveIp4Callback

```
typedef void(* otNat64ReceiveIp4Callback) (otMessage *aMessage, void *aContext) )(otMessage *aMessage, void *aContext)
```

Pointer is called when an IPv4 datagram (translated by NAT64 translator) is received.

Parameters

[in]	aMessage	A pointer to the message buffer containing the received IPv6 datagram. This function transfers the ownership of the `aMessage` to the receiver of the callback. The message should be freed by the receiver of the callback after it is processed.
[in]	aContext	A pointer to application-specific context.

Definition at line 369 of file include/openthread/nat64.h

Variable Documentation

OT_TOOL_PACKED_END

```
OT_TOOL_PACKED_BEGIN struct otIp4Address OT_TOOL_PACKED_END
```

Definition at line `71` of file `include/openthread/nat64.h`

Function Documentation

otNat64GetCounters

```
void otNat64GetCounters (otInstance *aInstance, otNat64ProtocolCounters *aCounters)
```

Gets NAT64 translator counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aCounters	A pointer to an `otNat64Counters` where the counters of NAT64 translator will be placed.

The counter is counted since the instance initialized.

Available when `OPENTHREAD_CONFIG_NAT64_TRANSLATOR_ENABLE` is enabled.

Definition at line `150` of file `include/openthread/nat64.h`

otNat64GetErrorCounters

```
void otNat64GetErrorCounters (otInstance *aInstance, otNat64ErrorCounters *aCounters)
```

Gets the NAT64 translator error counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aCounters	A pointer to an `otNat64Counters` where the counters of NAT64 translator will be placed.

The counters are initialized to zero when the OpenThread instance is initialized.

Definition at line `161` of file `include/openthread/nat64.h`

otNat64InitAddressMappingIterator

```
void otNat64InitAddressMappingIterator (otInstance *aInstance, otNat64AddressMappingIterator *aIterator)
```

Initializes an `otNat64AddressMappingIterator`.

Parameters

[in]	aInstance	The OpenThread instance.
[out]	aIterator	A pointer to the iterator to initialize.

An iterator MUST be initialized before it is used.

An iterator can be initialized again to restart from the beginning of the mapping info.

Definition at line `206` of file `include/openthread/nat64.h`

otNat64GetNextAddressMapping

> otError otNat64GetNextAddressMapping (otInstance *aInstance, otNat64AddressMappingIterator *aIterator, otNat64AddressMapping *aMapping)

Gets the next AddressMapping info (using an iterator).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the iterator. On success the iterator will be updated to point to next NAT64 address mapping record. To get the first entry the iterator should be set to OT_NAT64_ADDRESS_MAPPING_ITERATOR_INIT.
[out]	aMapping	A pointer to an `otNat64AddressMapping` where information of next NAT64 address mapping record is placed (on success).

Available when `OPENTHREAD_CONFIG_NAT64_TRANSLATOR_ENABLE` is enabled.

Definition at line `225` of file `include/openthread/nat64.h`

otNat64GetTranslatorState

> otNat64State otNat64GetTranslatorState (otInstance *aInstance)

Gets the state of NAT64 translator.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Available when `OPENTHREAD_CONFIG_NAT64_TRANSLATOR_ENABLE` is enabled.

Definition at line `254` of file `include/openthread/nat64.h`

otNat64GetPrefixManagerState

> otNat64State otNat64GetPrefixManagerState (otInstance *aInstance)

Gets the state of NAT64 prefix manager.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Available when `OPENTHREAD_CONFIG_NAT64_BORDER_ROUTING_ENABLE` is enabled.

Definition at line `273` of file `include/openthread/nat64.h`

otNat64SetEnabled

> void otNat64SetEnabled (otInstance *aInstance, bool aEnabled)

Enable or disable NAT64 functions.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	A boolean to enable/disable the NAT64 functions

Note: This includes the NAT64 Translator (when `OPENTHREAD_CONFIG_NAT64_TRANSLATOR_ENABLE` is enabled) and the NAT64 Prefix Manager (when `OPENTHREAD_CONFIG_NAT64_BORDER_ROUTING_ENABLE` is enabled).

When `OPENTHREAD_CONFIG_NAT64_TRANSLATOR_ENABLE` is enabled, setting disabled to true resets the mapping table in the translator.

Available when `OPENTHREAD_CONFIG_NAT64_TRANSLATOR_ENABLE` or `OPENTHREAD_CONFIG_NAT64_BORDER_ROUTING_ENABLE` is enabled.

See Also

- otNat64GetTranslatorState
- otNat64GetPrefixManagerState

Definition at line `294` of file `include/openthread/nat64.h`

otIp4NewMessage

> otMessage * otIp4NewMessage (otInstance *aInstance, const otMessageSettings *aSettings)

Allocate a new message buffer for sending an IPv4 message to the NAT64 translator.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSettings	A pointer to the message settings or NULL to set default settings.

Message buffers allocated by this function will have 20 bytes (difference between the size of IPv6 headers and IPv4 header sizes) reserved.

Available when `OPENTHREAD_CONFIG_NAT64_TRANSLATOR_ENABLE` is enabled.

Note

- If `aSettings` is `NULL`, the link layer security is enabled and the message priority is set to OT_MESSAGE_PRIORITY_NORMAL by default.

Returns

- A pointer to the message buffer or NULL if no message buffers are available or parameters are invalid.

See Also

- otNat64Send

Definition at line `315` of file `include/openthread/nat64.h`

otNat64SetIp4Cidr

> otError otNat64SetIp4Cidr (otInstance *aInstance, const otIp4Cidr *aCidr)

Sets the CIDR used when setting the source address of the outgoing translated IPv4 packets.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCidr	A pointer to an otIp4Cidr for the IPv4 CIDR block for NAT64.

Is available only when OPENTHREAD_CONFIG_NAT64_TRANSLATOR_ENABLE is enabled.

Note

- A valid CIDR must have a non-zero prefix length. The actual addresses pool is limited by the size of the mapping pool and the number of addresses available in the CIDR block.
- This function can be called at any time, but the NAT64 translator will be reset and all existing sessions will be expired when updating the configured CIDR.

See Also

- otBorderRouterSend
- otBorderRouterSetReceiveCallback

Definition at line `338` of file `include/openthread/nat64.h`

otNat64Send

```
otError otNat64Send (otInstance *aInstance, otMessage *aMessage)
```

Translates an IPv4 datagram to an IPv6 datagram and sends via the Thread interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the message buffer containing the IPv4 datagram.

The caller transfers ownership of `aMessage` when making this call. OpenThread will free `aMessage` when processing is complete, including when a value other than `OT_ERROR_NONE` is returned.

Definition at line `358` of file `include/openthread/nat64.h`

otNat64SetReceiveIp4Callback

```
void otNat64SetReceiveIp4Callback (otInstance *aInstance, otNat64ReceiveIp4Callback aCallback, void *aContext)
```

Registers a callback to provide received IPv4 datagrams.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function that is called when an IPv4 datagram is received or NULL to disable the callback.
[in]	aContext	A pointer to application-specific context.

Definition at line `380` of file `include/openthread/nat64.h`

otNat64GetCidr

```
otError otNat64GetCidr (otInstance *aInstance, otIp4Cidr *aCidr)
```

Gets the IPv4 CIDR configured in the NAT64 translator.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aCidr	A pointer to an otIp4Cidr. Where the CIDR will be filled.

Available when `OPENTHREAD_CONFIG_NAT64_TRANSLATOR_ENABLE` is enabled.

Definition at line `391` of file `include/openthread/nat64.h`

otIp4IsAddressEqual

```
bool otIp4IsAddressEqual (const otIp4Address *aFirst, const otIp4Address *aSecond)
```

Test if two IPv4 addresses are the same.

Parameters

[in]	aFirst	A pointer to the first IPv4 address to compare.
[in]	aSecond	A pointer to the second IPv4 address to compare.

Definition at line `403` of file `include/openthread/nat64.h`

otIp4ExtractFromIp6Address

```
void otIp4ExtractFromIp6Address (uint8_t aPrefixLength, const otIp6Address *aIp6Address, otIp4Address *aIp4Address)
```

Set `aIp4Address` by performing NAT64 address translation from `aIp6Address` as specified in RFC 6052.

Parameters

[in]	aPrefixLength	The prefix length to use for IPv4/IPv6 translation.
[in]	aIp6Address	A pointer to an IPv6 address.
[out]	aIp4Address	A pointer to output the IPv4 address.

The NAT64 `aPrefixLength` MUST be one of the following values: 32, 40, 48, 56, 64, or 96, otherwise the behavior of this method is undefined.

Definition at line `417` of file `include/openthread/nat64.h`

otIp4AddressToString

```
void otIp4AddressToString (const otIp4Address *aAddress, char *aBuffer, uint16_t aSize)
```

Converts the address to a string.

Parameters

[in]	aAddress	A pointer to an IPv4 address (MUST NOT be NULL).
[out]	aBuffer	A pointer to a char array to output the string (MUST NOT be `nullptr`).
[in]	aSize	The size of `aBuffer` (in bytes).

The string format uses quad-dotted notation of four bytes in the address (e.g., "127.0.0.1").

If the resulting string does not fit in `aBuffer` (within its `aSize` characters), the string will be truncated but the outputted string is always null-terminated.

Definition at line `434` of file `include/openthread/nat64.h`

otIp4CidrFromString

```
otError otIp4CidrFromString (const char *aString, otIp4Cidr *aCidr)
```

Converts a human-readable IPv4 CIDR string into a binary representation.

Parameters

[in]	aString	A pointer to a NULL-terminated string.
[out]	aCidr	A pointer to an IPv4 CIDR.

Definition at line `448` of file `include/openthread/nat64.h`

otIp4CidrToString

```
void otIp4CidrToString (const otIp4Cidr *aCidr, char *aBuffer, uint16_t aSize)
```

Converts the IPv4 CIDR to a string.

Parameters

[in]	aCidr	A pointer to an IPv4 CIDR (MUST NOT be NULL).
[out]	aBuffer	A pointer to a char array to output the string (MUST NOT be `nullptr`).
[in]	aSize	The size of `aBuffer` (in bytes).

The string format uses quad-dotted notation of four bytes in the address with the length of prefix (e.g., "127.0.0.1/32").

If the resulting string does not fit in `aBuffer` (within its `aSize` characters), the string will be truncated but the outputted string is always null-terminated.

Definition at line `464` of file `include/openthread/nat64.h`

otIp4AddressFromString

```
otError otIp4AddressFromString (const char *aString, otIp4Address *aAddress)
```

Converts a human-readable IPv4 address string into a binary representation.

Parameters

[in]	aString	A pointer to a NULL-terminated string.
[out]	aAddress	A pointer to an IPv4 address.

Definition at line `476` of file `include/openthread/nat64.h`

otNat64SynthesizeIp6Address

```
otError otNat64SynthesizeIp6Address (otInstance *aInstance, const otIp4Address *aIp4Address, otIp6Address *aIp6Address)
```

Sets the IPv6 address by performing NAT64 address translation from the preferred NAT64 prefix and the given IPv4 address as specified in RFC 6052.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aIp4Address	A pointer to the IPv4 address to translate to IPv6.
[out]	aIp6Address	A pointer to the synthesized IPv6 address.

Returns

- OT_ERROR_NONE Successfully synthesized the IPv6 address from NAT64 prefix and IPv4 address.
- OT_ERROR_INVALID_STATE No valid NAT64 prefix in the network data.

Definition at line `490` of file `include/openthread/nat64.h`

Macro Definition Documentation

OT_IP4_ADDRESS_SIZE

```
#define OT_IP4_ADDRESS_SIZE
```

Value:

```
4
```

Size of an IPv4 address (bytes)

Definition at line `55` of file `include/openthread/nat64.h`

OT_IP4_ADDRESS_STRING_SIZE

```
#define OT_IP4_ADDRESS_STRING_SIZE
```

Value:

```
17
```

Length of 000.000.000.000 plus a suffix NUL.

Definition at line `419` of file `include/openthread/nat64.h`

OT_IP4_CIDR_STRING_SIZE

```
#define OT_IP4_CIDR_STRING_SIZE
```

Value:

```
20
```

Length of 000.000.000.000/00 plus a suffix NUL.

Definition at line 436 of file include/openthread/nat64.h

otIp4Address

Represents an IPv4 address.

Modules

otIp4Address::OT_TOOL_PACKED_FIELD

Public Attributes

union otIp4Address::OT_TOOL_PACKED_FIELD	mFields

Public Attribute Documentation

mFields

```
union otIp4Address::OT_TOOL_PACKED_FIELD otIp4Address::mFields
```

Definition at line 70 of file include/openthread/nat64.h

otIp4Address

Public Attributes

uint8_t	m8 8-bit fields
uint32_t	m32 32-bit representation

Public Attribute Documentation

m8

```
uint8_t otIp4Address::OT_TOOL_PACKED_FIELD::m8[OT_IP4_ADDRESS_SIZE]
```

8-bit fields

Definition at line 68 of file include/openthread/nat64.h

m32

```
uint32_t otIp4Address::OT_TOOL_PACKED_FIELD::m32
```

32-bit representation

Definition at line 69 of file include/openthread/nat64.h

otIp4Cidr

Represents an IPv4 CIDR block.

Public Attributes

otIp4Address	mAddress
uint8_t	mLength

Public Attribute Documentation

mAddress

> otIp4Address otIp4Cidr::mAddress

Definition at line 87 of file include/openthread/nat64.h

mLength

> uint8_t otIp4Cidr::mLength

Definition at line 88 of file include/openthread/nat64.h

otNat64Counters

Represents the counters for NAT64.

Public Attributes

uint64_t	m4To6Packets Number of packets translated from IPv4 to IPv6.
uint64_t	m4To6Bytes Sum of size of packets translated from IPv4 to IPv6.
uint64_t	m6To4Packets Number of packets translated from IPv6 to IPv4.
uint64_t	m6To4Bytes Sum of size of packets translated from IPv6 to IPv4.

Public Attribute Documentation

m4To6Packets

```
uint64_t otNat64Counters::m4To6Packets
```

Number of packets translated from IPv4 to IPv6.

Definition at line `97` of file `include/openthread/nat64.h`

m4To6Bytes

```
uint64_t otNat64Counters::m4To6Bytes
```

Sum of size of packets translated from IPv4 to IPv6.

Definition at line `98` of file `include/openthread/nat64.h`

m6To4Packets

```
uint64_t otNat64Counters::m6To4Packets
```

Number of packets translated from IPv6 to IPv4.

Definition at line `99` of file `include/openthread/nat64.h`

m6To4Bytes

```
uint64_t otNat64Counters::m6To4Bytes
```

Sum of size of packets translated from IPv6 to IPv4.

Definition at line 100 of file include/openthread/nat64.h

otNat64ProtocolCounters

Represents the counters for the protocols supported by NAT64.

Public Attributes

otNat64Counters	mTotal Counters for sum of all protocols.
otNat64Counters	mIcmp Counters for ICMP and ICMPv6.
otNat64Counters	mUdp Counters for UDP.
otNat64Counters	mTcp Counters for TCP.

Public Attribute Documentation

mTotal

```
otNat64Counters otNat64ProtocolCounters::mTotal
```

Counters for sum of all protocols.

Definition at line `109` of file `include/openthread/nat64.h`

mIcmp

```
otNat64Counters otNat64ProtocolCounters::mIcmp
```

Counters for ICMP and ICMPv6.

Definition at line `110` of file `include/openthread/nat64.h`

mUdp

```
otNat64Counters otNat64ProtocolCounters::mUdp
```

Counters for UDP.

Definition at line `111` of file `include/openthread/nat64.h`

mTcp

```
otNat64Counters otNat64ProtocolCounters::mTcp
```

Counters for TCP.

Definition at line 112 of file include/openthread/nat64.h

otNat64ErrorCounters

Represents the counters of dropped packets due to errors when handling NAT64 packets.

Public Attributes

uint64_t	mCount4To6 Errors translating IPv4 packets.
uint64_t	mCount6To4 Errors translating IPv6 packets.

Public Attribute Documentation

mCount4To6

```
uint64_t otNat64ErrorCounters::mCount4To6[OT_NAT64_DROP_REASON_COUNT]
```

Errors translating IPv4 packets.

Definition at line 135 of file include/openthread/nat64.h

mCount6To4

```
uint64_t otNat64ErrorCounters::mCount6To4[OT_NAT64_DROP_REASON_COUNT]
```

Errors translating IPv6 packets.

Definition at line 136 of file include/openthread/nat64.h

otNat64AddressMapping

Represents an address mapping record for NAT64.

Note

- The counters will be reset for each mapping session even for the same address pair. Applications can use `mId` to identify different sessions to calculate the packets correctly.

Public Attributes

uint64_t	mId The unique id for a mapping session.
otIp4Address	mIp4 The IPv4 address of the mapping.
otIp6Address	mIp6 The IPv6 address of the mapping.
uint32_t	mRemainingTimeMs Remaining time before expiry in milliseconds.
otNat64ProtocolCounters	mCounters

Public Attribute Documentation

mId

```
uint64_t otNat64AddressMapping::mId
```

The unique id for a mapping session.

Definition at line `172` of file `include/openthread/nat64.h`

mIp4

```
otIp4Address otNat64AddressMapping::mIp4
```

The IPv4 address of the mapping.

Definition at line `174` of file `include/openthread/nat64.h`

mIp6

```
otIp6Address otNat64AddressMapping::mIp6
```

The IPv6 address of the mapping.

Definition at line `175` of file `include/openthread/nat64.h`

mRemainingTimeMs

```
uint32_t otNat64AddressMapping::mRemainingTimeMs
```

Remaining time before expiry in milliseconds.

Definition at line 176 of file include/openthread/nat64.h

mCounters

```
otNat64ProtocolCounters otNat64AddressMapping::mCounters
```

Definition at line 178 of file include/openthread/nat64.h

otNat64AddressMappingIterator

Used to iterate through NAT64 address mappings.

The fields in this type are opaque (intended for use by OpenThread core only) and therefore should not be accessed or used by caller.

Before using an iterator, it MUST be initialized using `otNat64AddressMappingIteratorInit()`.

Public Attributes

void *	mPtr

Public Attribute Documentation

mPtr

```
void* otNat64AddressMappingIterator::mPtr
```

Definition at line `192` of file `include/openthread/nat64.h`

SRP

SRP

This module includes functions that control SRP client behavior.

This module includes functions of the Service Registration Protocol.

This module includes functions for SRP client buffers and service pool.

Functions in this module are only available when feature OPENTHREAD_CONFIG_SRP_CLIENT_BUFFERS_ENABLE is enabled.

Modules

otSrpClientHostInfo

otSrpClientService

otSrpClientBuffersServiceEntry

otSrpServerTtlConfig

otSrpServerLeaseConfig

otSrpServerLeaseInfo

otSrpServerResponseCounters

Enumerations

enum otSrpClientItemState {

OT_SRP_CLIENT_ITEM_STATE_TO_ADD
OT_SRP_CLIENT_ITEM_STATE_ADDING
OT_SRP_CLIENT_ITEM_STATE_TO_REFRESH
OT_SRP_CLIENT_ITEM_STATE_REFRESHING
OT_SRP_CLIENT_ITEM_STATE_TO_REMOVE
OT_SRP_CLIENT_ITEM_STATE_REMOVING
OT_SRP_CLIENT_ITEM_STATE_REGISTERED
OT_SRP_CLIENT_ITEM_STATE_REMOVED

}

Specifies an SRP client item (service or host info) state.

enum otSrpServerState {

OT_SRP_SERVER_STATE_DISABLED = 0
OT_SRP_SERVER_STATE_RUNNING = 1
OT_SRP_SERVER_STATE_STOPPED = 2

}

Represents the state of the SRP server.

enum otSrpServerAddressMode {

OT_SRP_SERVER_ADDRESS_MODE_UNICAST = 0
OT_SRP_SERVER_ADDRESS_MODE_ANYCAST = 1

}

Represents the address mode used by the SRP server.

Typedefs

typedef struct otSrpClientHostInfo
Represents an SRP client host info.

typedef struct otSrpClientService
Represents an SRP client service.

typedef void(* otSrpClientCallback)(otError aError, const otSrpClientHostInfo *aHostInfo, const otSrpClientService *aServices, const otSrpClientService *aRemovedServices, void *aContext)
Pointer type defines the callback used by SRP client to notify user of changes/events/errors.

typedef void(* otSrpClientAutoStartCallback)(const otSockAddr *aServerSockAddr, void *aContext)
Pointer type defines the callback used by SRP client to notify user when it is auto-started or stopped.

typedef struct otSrpClientBuffersServiceEntry
Represents a SRP client service pool entry.

typedef struct otSrpServerHost
This opaque type represents a SRP service host.

typedef struct otSrpServerService
This opaque type represents a SRP service.

typedef uint32_t otSrpServerServiceUpdateId
The ID of a SRP service update transaction on the SRP Server.

typedef enum otSrpServerAddressMode
Represents the address mode used by the SRP server.

typedef struct otSrpServerTtlConfig
Includes SRP server TTL configurations.

typedef struct otSrpServerLeaseConfig
Includes SRP server LEASE and KEY-LEASE configurations.

typedef struct otSrpServerLeaseInfo
Includes SRP server lease information of a host/service.

typedef struct otSrpServerResponseCounters
Includes the statistics of SRP server responses.

typedef void(* otSrpServerServiceUpdateHandler)(otSrpServerServiceUpdateId aId, const otSrpServerHost *aHost, uint32_t aTimeout, void *aContext)
Handles SRP service updates.

Functions

otError otSrpClientStart(otInstance *aInstance, const otSockAddr *aServerSockAddr)
Starts the SRP client operation.

void otSrpClientStop(otInstance *aInstance)
Stops the SRP client operation.

bool	otSrpClientIsRunning(otInstance *aInstance) Indicates whether the SRP client is running or not.
const otSockAddr *	otSrpClientGetServerAddress(otInstance *aInstance) Gets the socket address (IPv6 address and port number) of the SRP server which is being used by SRP client.
void	otSrpClientSetCallback(otInstance *aInstance, otSrpClientCallback aCallback, void *aContext) Sets the callback to notify caller of events/changes from SRP client.
void	otSrpClientEnableAutoStartMode(otInstance *aInstance, otSrpClientAutoStartCallback aCallback, void *aContext) Enables the auto-start mode.
void	otSrpClientDisableAutoStartMode(otInstance *aInstance) Disables the auto-start mode.
bool	otSrpClientIsAutoStartModeEnabled(otInstance *aInstance) Indicates the current state of auto-start mode (enabled or disabled).
uint32_t	otSrpClientGetTtl(otInstance *aInstance) Gets the TTL value in every record included in SRP update requests.
void	otSrpClientSetTtl(otInstance *aInstance, uint32_t aTtl) Sets the TTL value in every record included in SRP update requests.
uint32_t	otSrpClientGetLeaseInterval(otInstance *aInstance) Gets the default lease interval used in SRP update requests.
void	otSrpClientSetLeaseInterval(otInstance *aInstance, uint32_t aInterval) Sets the default lease interval used in SRP update requests.
uint32_t	otSrpClientGetKeyLeaseInterval(otInstance *aInstance) Gets the default key lease interval used in SRP update requests.
void	otSrpClientSetKeyLeaseInterval(otInstance *aInstance, uint32_t aInterval) Sets the default key lease interval used in SRP update requests.
const otSrpClientHostInfo *	otSrpClientGetHostInfo(otInstance *aInstance) Gets the host info.
otError	otSrpClientSetHostName(otInstance *aInstance, const char *aName) Sets the host name label.
otError	otSrpClientEnableAutoHostAddress(otInstance *aInstance) Enables auto host address mode.
otError	otSrpClientSetHostAddresses(otInstance *aInstance, const otIp6Address *aIp6Addresses, uint8_t aNumAddresses) Sets/updates the list of host IPv6 address.
otError	otSrpClientAddService(otInstance *aInstance, otSrpClientService *aService) Adds a service to be registered with server.
otError	otSrpClientRemoveService(otInstance *aInstance, otSrpClientService *aService) Requests a service to be unregistered with server.
otError	otSrpClientClearService(otInstance *aInstance, otSrpClientService *aService) Clears a service, immediately removing it from the client service list.
const otSrpClientService *	otSrpClientGetServices(otInstance *aInstance) Gets the list of services being managed by client.

otError	otSrpClientRemoveHostAndServices(otInstance *aInstance, bool aRemoveKeyLease, bool aSendUnregToServer) Starts the remove process of the host info and all services.
void	otSrpClientClearHostAndServices(otInstance *aInstance) Clears all host info and all the services.
const char *	otSrpClientGetDomainName(otInstance *aInstance) Gets the domain name being used by SRP client.
otError	otSrpClientSetDomainName(otInstance *aInstance, const char *aName) Sets the domain name to be used by SRP client.
const char *	otSrpClientItemStateToString(otSrpClientItemState aItemState) Converts a `otSrpClientItemState` to a string.
void	otSrpClientSetServiceKeyRecordEnabled(otInstance *aInstance, bool aEnabled) Enables/disables "service key record inclusion" mode.
bool	otSrpClientIsServiceKeyRecordEnabled(otInstance *aInstance) Indicates whether the "service key record inclusion" mode is enabled or disabled.
char *	otSrpClientBuffersGetHostNameString(otInstance *aInstance, uint16_t *aSize) Gets the string buffer to use for SRP client host name.
otIp6Address *	otSrpClientBuffersGetHostAddressesArray(otInstance *aInstance, uint8_t *aArrayLength) Gets the array of IPv6 address entries to use as SRP client host address list.
otSrpClientBuffersServiceEntry *	otSrpClientBuffersAllocateService(otInstance *aInstance) Allocates a new service entry from the pool.
void	otSrpClientBuffersFreeService(otInstance *aInstance, otSrpClientBuffersServiceEntry *aService) Frees a previously allocated service entry.
void	otSrpClientBuffersFreeAllServices(otInstance *aInstance) Frees all previously allocated service entries.
char *	otSrpClientBuffersGetServiceEntryServiceNameString(otSrpClientBuffersServiceEntry *aEntry, uint16_t *aSize) Gets the string buffer for service name from a service entry.
char *	otSrpClientBuffersGetServiceEntryInstanceNameString(otSrpClientBuffersServiceEntry *aEntry, uint16_t *aSize) Gets the string buffer for service instance name from a service entry.
uint8_t *	otSrpClientBuffersGetServiceEntryTxtBuffer(otSrpClientBuffersServiceEntry *aEntry, uint16_t *aSize) Gets the buffer for TXT record from a service entry.
const char **	otSrpClientBuffersGetSubTypeLabelsArray(otSrpClientBuffersServiceEntry *aEntry, uint16_t *aArrayLength) Gets the array for service subtype labels from the service entry.
const char *	otSrpServerGetDomain(otInstance *aInstance) Returns the domain authorized to the SRP server.
otError	otSrpServerSetDomain(otInstance *aInstance, const char *aDomain) Sets the domain on the SRP server.
otSrpServerState	otSrpServerGetState(otInstance *aInstance) Returns the state of the SRP server.
uint16_t	otSrpServerGetPort(otInstance *aInstance) Returns the port the SRP server is listening to.

otSrpServerAddressMode	otSrpServerGetAddressMode(otInstance *aInstance) Returns the address mode being used by the SRP server.
otError	otSrpServerSetAddressMode(otInstance *aInstance, otSrpServerAddressMode aMode) Sets the address mode to be used by the SRP server.
uint8_t	otSrpServerGetAnycastModeSequenceNumber(otInstance *aInstance) Returns the sequence number used with anycast address mode.
otError	otSrpServerSetAnycastModeSequenceNumber(otInstance *aInstance, uint8_t aSequenceNumber) Sets the sequence number used with anycast address mode.
void	otSrpServerSetEnabled(otInstance *aInstance, bool aEnabled) Enables/disables the SRP server.
void	otSrpServerSetAutoEnableMode(otInstance *aInstance, bool aEnabled) Enables/disables the auto-enable mode on SRP server.
bool	otSrpServerIsAutoEnableMode(otInstance *aInstance) Indicates whether the auto-enable mode is enabled or disabled.
void	otSrpServerGetTtlConfig(otInstance *aInstance, otSrpServerTtlConfig *aTtlConfig) Returns SRP server TTL configuration.
otError	otSrpServerSetTtlConfig(otInstance *aInstance, const otSrpServerTtlConfig *aTtlConfig) Sets SRP server TTL configuration.
void	otSrpServerGetLeaseConfig(otInstance *aInstance, otSrpServerLeaseConfig *aLeaseConfig) Returns SRP server LEASE and KEY-LEASE configurations.
otError	otSrpServerSetLeaseConfig(otInstance *aInstance, const otSrpServerLeaseConfig *aLeaseConfig) Sets SRP server LEASE and KEY-LEASE configurations.
void	otSrpServerSetServiceUpdateHandler(otInstance *aInstance, otSrpServerServiceUpdateHandler aServiceHandler, void *aContext) Sets the SRP service updates handler on SRP server.
void	otSrpServerHandleServiceUpdateResult(otInstance *aInstance, otSrpServerServiceUpdateId aId, otError aError) Reports the result of processing a SRP update to the SRP server.
const otSrpServerHost *	otSrpServerGetNextHost(otInstance *aInstance, const otSrpServerHost *aHost) Returns the next registered host on the SRP server.
const otSrpServerResponseCounters *	otSrpServerGetResponseCounters(otInstance *aInstance) Returns the response counters of the SRP server.
bool	otSrpServerHostIsDeleted(const otSrpServerHost *aHost) Tells if the SRP service host has been deleted.
const char *	otSrpServerHostGetFullName(const otSrpServerHost *aHost) Returns the full name of the host.
bool	otSrpServerHostMatchesFullName(const otSrpServerHost *aHost, const char *aFullName) Indicates whether the host matches a given host name.
const otIp6Address *	otSrpServerHostGetAddresses(const otSrpServerHost *aHost, uint8_t *aAddressesNum) Returns the addresses of given host.
void	otSrpServerHostGetLeaseInfo(const otSrpServerHost *aHost, otSrpServerLeaseInfo *aLeaseInfo) Returns the LEASE and KEY-LEASE information of a given host.

const otSrpServerService *	otSrpServerHostGetNextService(const otSrpServerHost *aHost, const otSrpServerService *aService) Returns the next service of given host.
bool	otSrpServerServiceIsDeleted(const otSrpServerService *aService) Indicates whether or not the SRP service has been deleted.
const char *	otSrpServerServiceGetInstanceName(const otSrpServerService *aService) Returns the full service instance name of the service.
bool	otSrpServerServiceMatchesInstanceName(const otSrpServerService *aService, const char *aInstanceName) Indicates whether this service matches a given service instance name.
const char *	otSrpServerServiceGetInstanceLabel(const otSrpServerService *aService) Returns the service instance label (first label in instance name) of the service.
const char *	otSrpServerServiceGetServiceName(const otSrpServerService *aService) Returns the full service name of the service.
bool	otSrpServerServiceMatchesServiceName(const otSrpServerService *aService, const char *aServiceName) Indicates whether this service matches a given service name.
uint16_t	otSrpServerServiceGetNumberOfSubTypes(const otSrpServerService *aService) Gets the number of sub-types of the service.
const char *	otSrpServerServiceGetSubTypeServiceNameAt(const otSrpServerService *aService, uint16_t aIndex) Gets the sub-type service name (full name) of the service at a given index.
bool	otSrpServerServiceHasSubTypeServiceName(const otSrpServerService *aService, const char *aSubTypeServiceName) Indicates whether or not the service has a given sub-type.
otError	otSrpServerParseSubTypeServiceName(const char *aSubTypeServiceName, char *aLabel, uint8_t aLabelSize) Parses a sub-type service name (full name) and extracts the sub-type label.
uint16_t	otSrpServerServiceGetPort(const otSrpServerService *aService) Returns the port of the service instance.
uint16_t	otSrpServerServiceGetWeight(const otSrpServerService *aService) Returns the weight of the service instance.
uint16_t	otSrpServerServiceGetPriority(const otSrpServerService *aService) Returns the priority of the service instance.
uint32_t	otSrpServerServiceGetTtl(const otSrpServerService *aService) Returns the TTL of the service instance.
const uint8_t *	otSrpServerServiceGetTxtData(const otSrpServerService *aService, uint16_t *aDataLength) Returns the TXT record data of the service instance.
const otSrpServerHost *	otSrpServerServiceGetHost(const otSrpServerService *aService) Returns the host which the service instance reside on.
void	otSrpServerServiceGetLeaseInfo(const otSrpServerService *aService, otSrpServerLeaseInfo *aLeaseInfo) Returns the LEASE and KEY-LEASE information of a given service.

Enumeration Documentation

otSrpClientItemState

> otSrpClientItemState

Specifies an SRP client item (service or host info) state.

Enumerator	
OT_SRP_CLIENT_ITEM_STATE_TO_ADD	Item to be added/registered.
OT_SRP_CLIENT_ITEM_STATE_ADDING	Item is being added/registered.
OT_SRP_CLIENT_ITEM_STATE_TO_REFRESH	Item to be refreshed (re-register to renew lease).
OT_SRP_CLIENT_ITEM_STATE_REFRESHING	Item is being refreshed.
OT_SRP_CLIENT_ITEM_STATE_TO_REMOVE	Item to be removed.
OT_SRP_CLIENT_ITEM_STATE_REMOVING	Item is being removed.
OT_SRP_CLIENT_ITEM_STATE_REGISTERED	Item is registered with server.
OT_SRP_CLIENT_ITEM_STATE_REMOVED	Item is removed.

Definition at line `59` of file `include/openthread/srp_client.h`

otSrpServerState

> otSrpServerState

Represents the state of the SRP server.

Enumerator	
OT_SRP_SERVER_STATE_DISABLED	The SRP server is disabled.
OT_SRP_SERVER_STATE_RUNNING	The SRP server is enabled and running.
OT_SRP_SERVER_STATE_STOPPED	The SRP server is enabled but stopped.

Definition at line `80` of file `include/openthread/srp_server.h`

otSrpServerAddressMode

> otSrpServerAddressMode

Represents the address mode used by the SRP server.

Address mode specifies how the address and port number are determined by the SRP server and how this info is published in the Thread Network Data.

Enumerator	
OT_SRP_SERVER_ADDRESS_MODE_UNICAST	Unicast address mode.
OT_SRP_SERVER_ADDRESS_MODE_ANYCAST	Anycast address mode.

Definition at line `94` of file `include/openthread/srp_server.h`

Typedef Documentation

otSrpClientHostInfo

> typedef struct otSrpClientHostInfo otSrpClientHostInfo

Represents an SRP client host info.

Definition at line `82` of file `include/openthread/srp_client.h`

otSrpClientService

```
typedef struct otSrpClientService otSrpClientService
```

Represents an SRP client service.

The values in this structure, including the string buffers for the names and the TXT record entries, MUST persist and stay constant after an instance of this structure is passed to OpenThread from `otSrpClientAddService()` or `otSrpClientRemoveService()`.

The `mState`, `mData`, `mNext` fields are used/managed by OT core only. Their value is ignored when an instance of `otSrpClientService` is passed in `otSrpClientAddService()` or `otSrpClientRemoveService()` or other functions. The caller does not need to set these fields.

The `mLease` and `mKeyLease` fields specify the desired lease and key lease intervals for this service. Zero value indicates that the interval is unspecified and then the default lease or key lease intervals from `otSrpClientGetLeaseInterval()` and `otSrpClientGetKeyLeaseInterval()` are used for this service. If the key lease interval (whether set explicitly or determined from the default) is shorter than the lease interval for a service, SRP client will re-use the lease interval value for key lease interval as well. For example, if in service `mLease` is explicitly set to 2 days and `mKeyLease` is set to zero and default key lease is set to 1 day, then when registering this service, the requested key lease for this service is also set to 2 days.

Definition at line `119` of file `include/openthread/srp_client.h`

otSrpClientCallback

```
typedef void(* otSrpClientCallback) (otError aError, const otSrpClientHostInfo *aHostInfo, const otSrpClientService *aServices, const otSrpClientService *aRemovedServices, void *aContext) )(otError aError, const otSrpClientHostInfo *aHostInfo, const otSrpClientService *aServices, const otSrpClientService *aRemovedServices, void *aContext)
```

Pointer type defines the callback used by SRP client to notify user of changes/events/errors.

Parameters

[in]	aError	The error (see above).
[in]	aHostInfo	A pointer to host info.
[in]	aServices	The head of linked-list containing all services (excluding the ones removed). NULL if the list is empty.
[in]	aRemovedServices	The head of linked-list containing all removed services. NULL if the list is empty.
[in]	aContext	A pointer to an arbitrary context (provided when callback was registered).

This callback is invoked on a successful registration of an update (i.e., add/remove of host-info and/or some service(s)) with the SRP server, or if there is a failure or error (e.g., server rejects a update request or client times out waiting for response, etc).

In case of a successful reregistration of an update, `aError` parameter would be `OT_ERROR_NONE` and the host info and the full list of services is provided as input parameters to the callback. Note that host info and services each track its own state in the corresponding `mState` member variable of the related data structure (the state indicating whether the host-info/service is registered or removed or still being added/removed, etc).

The list of removed services is passed as its own linked-list `aRemovedServices` in the callback. Note that when the callback is invoked, the SRP client (OpenThread implementation) is done with the removed service instances listed in `aRemovedServices` and no longer tracks/stores them (i.e., if from the callback we call `otSrpClientGetServices()` the removed

services will not be present in the returned list). Providing a separate list of removed services in the callback helps indicate to user which items are now removed and allow user to re-claim/reuse the instances.

If the server rejects an SRP update request, the DNS response code (RFC 2136) is mapped to the following errors:

- (0) NOERROR Success (no error condition) -> OT_ERROR_NONE
- (1) FORMERR Server unable to interpret due to format error -> OT_ERROR_PARSE
- (2) SERVFAIL Server encountered an internal failure -> OT_ERROR_FAILED
- (3) NXDOMAIN Name that ought to exist, does not exist -> OT_ERROR_NOT_FOUND
- (4) NOTIMP Server does not support the query type (OpCode) -> OT_ERROR_NOT_IMPLEMENTED
- (5) REFUSED Server refused for policy/security reasons -> OT_ERROR_SECURITY
- (6) YXDOMAIN Some name that ought not to exist, does exist -> OT_ERROR_DUPLICATED
- (7) YXRRSET Some RRset that ought not to exist, does exist -> OT_ERROR_DUPLICATED
- (8) NXRRSET Some RRset that ought to exist, does not exist -> OT_ERROR_NOT_FOUND
- (9) NOTAUTH Service is not authoritative for zone -> OT_ERROR_SECURITY
- (10) NOTZONE A name is not in the zone -> OT_ERROR_PARSE
- (20) BADNAME Bad name -> OT_ERROR_PARSE
- (21) BADALG Bad algorithm -> OT_ERROR_SECURITY
- (22) BADTRUN Bad truncation -> OT_ERROR_PARSE
- Other response codes -> OT_ERROR_FAILED

The following errors are also possible:

- OT_ERROR_RESPONSE_TIMEOUT : Timed out waiting for response from server (client would continue to retry).
- OT_ERROR_INVALID_ARGS : The provided service structure is invalid (e.g., bad service name or `otDnsTxtEntry`).
- OT_ERROR_NO_BUFS : Insufficient buffer to prepare or send the update message.

Note that in case of any failure, the client continues the operation, i.e. it prepares and (re)transmits the SRP update message to the server, after some wait interval. The retry wait interval starts from the minimum value and is increased by the growth factor every failure up to the max value (please see configuration parameter `OPENTHREAD_CONFIG_SRP_CLIENT_MIN_RETRY_WAIT_INTERVAL` and the related ones for more details).

Definition at line `176` of file `include/openthread/srp_client.h`

otSrpClientAutoStartCallback

```
typedef void(* otSrpClientAutoStartCallback) (const otSockAddr *aServerSockAddr, void *aContext) )(const otSockAddr *aServerSockAddr, void *aContext)
```

Pointer type defines the callback used by SRP client to notify user when it is auto-started or stopped.

Parameters

[in]	aServerSockAddr	A non-NULL pointer indicates SRP server was started and pointer will give the selected server socket address. A NULL pointer indicates SRP server was stopped.
[in]	aContext	A pointer to an arbitrary context (provided when callback was registered).

This is only used when auto-start feature `OPENTHREAD_CONFIG_SRP_CLIENT_AUTO_START_API_ENABLE` is enabled.

This callback is invoked when auto-start mode is enabled and the SRP client is either automatically started or stopped.

Definition at line `195` of file `include/openthread/srp_client.h`

otSrpClientBuffersServiceEntry

```
typedef struct otSrpClientBuffersServiceEntry otSrpClientBuffersServiceEntry
```

Represents a SRP client service pool entry.

Definition at line 65 of file include/openthread/srp_client_buffers.h

otSrpServerHost

```
typedef struct otSrpServerHost otSrpServerHost
```

This opaque type represents a SRP service host.

Definition at line 62 of file include/openthread/srp_server.h

otSrpServerService

```
typedef struct otSrpServerService otSrpServerService
```

This opaque type represents a SRP service.

Definition at line 68 of file include/openthread/srp_server.h

otSrpServerServiceUpdateId

```
typedef uint32_t otSrpServerServiceUpdateId
```

The ID of a SRP service update transaction on the SRP Server.

Definition at line 74 of file include/openthread/srp_server.h

otSrpServerAddressMode

```
typedef enum otSrpServerAddressMode otSrpServerAddressMode
```

Represents the address mode used by the SRP server.

Address mode specifies how the address and port number are determined by the SRP server and how this info is published in the Thread Network Data.

Definition at line 98 of file include/openthread/srp_server.h

otSrpServerTtlConfig

```
typedef struct otSrpServerTtlConfig otSrpServerTtlConfig
```

Includes SRP server TTL configurations.

Definition at line 108 of file include/openthread/srp_server.h

otSrpServerLeaseConfig

```
typedef struct otSrpServerLeaseConfig otSrpServerLeaseConfig
```

Includes SRP server LEASE and KEY-LEASE configurations.

Definition at line `120` of file `include/openthread/srp_server.h`

otSrpServerLeaseInfo

```
typedef struct otSrpServerLeaseInfo otSrpServerLeaseInfo
```

Includes SRP server lease information of a host/service.

Definition at line `132` of file `include/openthread/srp_server.h`

otSrpServerResponseCounters

```
typedef struct otSrpServerResponseCounters otSrpServerResponseCounters
```

Includes the statistics of SRP server responses.

Definition at line `146` of file `include/openthread/srp_server.h`

otSrpServerServiceUpdateHandler

```
typedef void(* otSrpServerServiceUpdateHandler) (otSrpServerServiceUpdateId aId, const otSrpServerHost *aHost,
uint32_t aTimeout, void *aContext) )(otSrpServerServiceUpdateId aId, const otSrpServerHost *aHost, uint32_t aTimeout,
void *aContext)
```

Handles SRP service updates.

Parameters

[in]	aId	The service update transaction ID. This ID must be passed back with `otSrpServerHandleServiceUpdateResult`.
[in]	aHost	A pointer to the otSrpServerHost object which contains the SRP updates. The handler should publish/un-publish the host and each service points to this host with below rules: 1. If the host is not deleted (indicated by `otSrpServerHostIsDeleted`), then it should be published or updated with mDNS. Otherwise, the host should be un-published (remove AAAA RRs). 2. For each service points to this host, it must be un-published if the host is to be un-published. Otherwise, the handler should publish or update the service when it is not deleted (indicated by `otSrpServerServiceIsDeleted`) and un-publish it when deleted.
[in]	aTimeout	The maximum time in milliseconds for the handler to process the service event.
[in]	aContext	A pointer to application-specific context.

Is called by the SRP server to notify that a SRP host and possibly SRP services are being updated. It is important that the SRP updates are not committed until the handler returns the result by calling otSrpServerHandleServiceUpdateResult or times out after `aTimeout`.

A SRP service observer should always call otSrpServerHandleServiceUpdateResult with error code OT_ERROR_NONE immediately after receiving the update events.

A more generic handler may perform validations on the SRP host/services and rejects the SRP updates if any validation fails. For example, an Advertising Proxy should advertise (or remove) the host and services on a multicast-capable link and returns specific error code if any failure occurs.

See Also

- otSrpServerSetServiceUpdateHandler
- otSrpServerHandleServiceUpdateResult

Definition at line 373 of file include/openthread/srp_server.h

Function Documentation

otSrpClientStart

```
otError otSrpClientStart (otInstance *aInstance, const otSockAddr *aServerSockAddr)
```

Starts the SRP client operation.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aServerSockAddr	The socket address (IPv6 address and port number) of the SRP server.

SRP client will prepare and send "SRP Update" message to the SRP server once all the following conditions are met:

- The SRP client is started - otSrpClientStart() is called.
- Host name is set - otSrpClientSetHostName() is called.
- At least one host IPv6 address is set - otSrpClientSetHostName() is called.
- At least one service is added - otSrpClientAddService() is called.

It does not matter in which order these functions are called. When all conditions are met, the SRP client will wait for a short delay before preparing an "SRP Update" message and sending it to server. This delay allows for user to add multiple services and/or IPv6 addresses before the first SRP Update message is sent (ensuring a single SRP Update is sent containing all the info). The config OPENTHREAD_CONFIG_SRP_CLIENT_UPDATE_TX_DELAY specifies the delay interval.

Definition at line 222 of file include/openthread/srp_client.h

otSrpClientStop

```
void otSrpClientStop (otInstance *aInstance)
```

Stops the SRP client operation.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Stops any further interactions with the SRP server. Note that it does not remove or clear host info and/or list of services. It marks all services to be added/removed again once the client is (re)started.

Definition at line 233 of file include/openthread/srp_client.h

otSrpClientIsRunning

```
bool otSrpClientIsRunning (otInstance *aInstance)
```

Indicates whether the SRP client is running or not.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Returns

- TRUE if the SRP client is running, FALSE otherwise.

Definition at line `243` of file `include/openthread/srp_client.h`

otSrpClientGetServerAddress

```
const otSockAddr * otSrpClientGetServerAddress (otInstance *aInstance)
```

Gets the socket address (IPv6 address and port number) of the SRP server which is being used by SRP client.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

If the client is not running, the address is unspecified (all zero) with zero port number.

Returns

- A pointer to the SRP server's socket address (is always non-NULL).

Definition at line `256` of file `include/openthread/srp_client.h`

otSrpClientSetCallback

```
void otSrpClientSetCallback (otInstance *aInstance, otSrpClientCallback aCallback, void *aContext)
```

Sets the callback to notify caller of events/changes from SRP client.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aCallback	The callback to notify of events and changes. Can be NULL if not needed.
[in]	aContext	An arbitrary context used with `aCallback`.

The SRP client allows a single callback to be registered. So consecutive calls to this function will overwrite any previously set callback functions.

Definition at line `269` of file `include/openthread/srp_client.h`

otSrpClientEnableAutoStartMode

```
void otSrpClientEnableAutoStartMode (otInstance *aInstance, otSrpClientAutoStartCallback aCallback, void *aContext)
```

Enables the auto-start mode.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aCallback	A callback to notify when client is auto-started/stopped. Can be NULL if not needed.
[in]	aContext	A context to be passed when invoking `aCallback`.

This is only available when auto-start feature `OPENTHREAD_CONFIG_SRP_CLIENT_AUTO_START_API_ENABLE` is enabled.

Config option `OPENTHREAD_CONFIG_SRP_CLIENT_AUTO_START_DEFAULT_MODE` specifies the default auto-start mode (whether it is enabled or disabled at the start of OT stack).

When auto-start is enabled, the SRP client will monitor the Thread Network Data to discover SRP servers and select the preferred server and automatically start and stop the client when an SRP server is detected.

There are three categories of Network Data entries indicating presence of SRP sever. They are preferred in the following order:

1) Preferred unicast entries where server address is included in the service data. If there are multiple options, the one with numerically lowest IPv6 address is preferred.

2) Anycast entries each having a seq number. A larger sequence number in the sense specified by Serial Number Arithmetic logic in RFC-1982 is considered more recent and therefore preferred. The largest seq number using serial number arithmetic is preferred if it is well-defined (i.e., the seq number is larger than all other seq numbers). If it is not well-defined, then the numerically largest seq number is preferred.

3) Unicast entries where the server address info is included in server data. If there are multiple options, the one with numerically lowest IPv6 address is preferred.

When there is a change in the Network Data entries, client will check that the currently selected server is still present in the Network Data and is still the preferred one. Otherwise the client will switch to the new preferred server or stop if there is none.

When the SRP client is explicitly started through a successful call to `otSrpClientStart()`, the given SRP server address in `otSrpClientStart()` will continue to be used regardless of the state of auto-start mode and whether the same SRP server address is discovered or not in the Thread Network Data. In this case, only an explicit `otSrpClientStop()` call will stop the client.

Definition at line `310` of file `include/openthread/srp_client.h`

otSrpClientDisableAutoStartMode

```
void otSrpClientDisableAutoStartMode (otInstance *aInstance)
```

Disables the auto-start mode.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

This is only available when auto-start feature `OPENTHREAD_CONFIG_SRP_CLIENT_AUTO_START_API_ENABLE` is enabled.

Disabling the auto-start mode will not stop the client if it is already running but the client stops monitoring the Thread Network Data to verify that the selected SRP server is still present in it.

Note that a call to `otSrpClientStop()` will also disable the auto-start mode.

Definition at line `325` of file `include/openthread/srp_client.h`

otSrpClientIsAutoStartModeEnabled

```
bool otSrpClientIsAutoStartModeEnabled (otInstance *aInstance)
```

Indicates the current state of auto-start mode (enabled or disabled).

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

This is only available when auto-start feature `OPENTHREAD_CONFIG_SRP_CLIENT_AUTO_START_API_ENABLE` is enabled.

Returns

- TRUE if the auto-start mode is enabled, FALSE otherwise.

Definition at line `337` of file `include/openthread/srp_client.h`

otSrpClientGetTtl

```
uint32_t otSrpClientGetTtl (otInstance *aInstance)
```

Gets the TTL value in every record included in SRP update requests.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Note that this is the TTL requested by the SRP client. The server may choose to accept a different TTL.

By default, the TTL will equal the lease interval. Passing 0 or a value larger than the lease interval via `otSrpClientSetTtl()` will also cause the TTL to equal the lease interval.

Returns

- The TTL (in seconds).

Definition at line `352` of file `include/openthread/srp_client.h`

otSrpClientSetTtl

```
void otSrpClientSetTtl (otInstance *aInstance, uint32_t aTtl)
```

Sets the TTL value in every record included in SRP update requests.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aTtl	The TTL (in seconds). If value is zero or greater than lease interval, the TTL is set to the lease interval.

Changing the TTL does not impact the TTL of already registered services/host-info. It only affects future SRP update messages (i.e., adding new services and/or refreshes of the existing services).

Definition at line `365` of file `include/openthread/srp_client.h`

otSrpClientGetLeaseInterval

```
uint32_t otSrpClientGetLeaseInterval (otInstance *aInstance)
```

Gets the default lease interval used in SRP update requests.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

The default interval is used only for `otSrpClientService` instances with `mLease` set to zero.

Note that this is the lease duration requested by the SRP client. The server may choose to accept a different lease interval.

Returns

- The lease interval (in seconds).

Definition at line `380` of file `include/openthread/srp_client.h`

otSrpClientSetLeaseInterval

```
void otSrpClientSetLeaseInterval (otInstance *aInstance, uint32_t aInterval)
```

Sets the default lease interval used in SRP update requests.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aInterval	The lease interval (in seconds). If zero, the default value specified by `OPENTHREAD_CONFIG_SRP_CLIENT_DEFAULT_LEASE` would be used.

The default interval is used only for `otSrpClientService` instances with `mLease` set to zero.

Changing the lease interval does not impact the accepted lease interval of already registered services/host-info. It only affects any future SRP update messages (i.e., adding new services and/or refreshes of the existing services).

Definition at line `395` of file `include/openthread/srp_client.h`

otSrpClientGetKeyLeaseInterval

```
uint32_t otSrpClientGetKeyLeaseInterval (otInstance *aInstance)
```

Gets the default key lease interval used in SRP update requests.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

The default interval is used only for `otSrpClientService` instances with `mKeyLease` set to zero.

Note that this is the lease duration requested by the SRP client. The server may choose to accept a different lease interval.

Returns

- The key lease interval (in seconds).

Definition at line `410` of file `include/openthread/srp_client.h`

otSrpClientSetKeyLeaseInterval

```
void otSrpClientSetKeyLeaseInterval (otInstance *aInstance, uint32_t aInterval)
```

Sets the default key lease interval used in SRP update requests.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aInterval	The key lease interval (in seconds). If zero, the default value specified by OPENTHREAD_CONFIG_SRP_CLIENT_DEFAULT_KEY_LEASE would be used.

The default interval is used only for otSrpClientService instances with mKeyLease set to zero.

Changing the lease interval does not impact the accepted lease interval of already registered services/host-info. It only affects any future SRP update messages (i.e., adding new services and/or refreshes of existing services).

Definition at line 425 of file include/openthread/srp_client.h

otSrpClientGetHostInfo

```
const otSrpClientHostInfo * otSrpClientGetHostInfo (otInstance *aInstance)
```

Gets the host info.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Returns

- A pointer to host info structure.

Definition at line 435 of file include/openthread/srp_client.h

otSrpClientSetHostName

```
otError otSrpClientSetHostName (otInstance *aInstance, const char *aName)
```

Sets the host name label.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aName	A pointer to host name label string (MUST NOT be NULL). Pointer to the string buffer MUST persist and remain valid and constant after return from this function.

After a successful call to this function, otSrpClientCallback will be called to report the status of host info registration with SRP server.

The name string buffer pointed to by aName MUST persist and stay unchanged after returning from this function. OpenThread will keep the pointer to the string.

The host name can be set before client is started or after start but before host info is registered with server (host info should be in either STATE_TO_ADD or STATE_REMOVED).

Definition at line 458 of file include/openthread/srp_client.h

otSrpClientEnableAutoHostAddress

```
otError otSrpClientEnableAutoHostAddress (otInstance *aInstance)
```

Enables auto host address mode.

Parameters

N/A	aInstance	

When enabled host IPv6 addresses are automatically set by SRP client using all the unicast addresses on Thread netif excluding all link-local and mesh-local addresses. If there is no valid address, then Mesh Local EID address is added. The SRP client will automatically re-register when/if addresses on Thread netif are updated (new addresses are added or existing addresses are removed).

The auto host address mode can be enabled before start or during operation of SRP client except when the host info is being removed (client is busy handling a remove request from an call to `otSrpClientRemoveHostAndServices()` and host info still being in either `STATE_TO_REMOVE` or `STATE_REMOVING` states).

After auto host address mode is enabled, it can be disabled by a call to `otSrpClientSetHostAddresses()` which then explicitly sets the host addresses.

Definition at line `479` of file `include/openthread/srp_client.h`

otSrpClientSetHostAddresses

> otError otSrpClientSetHostAddresses (otInstance *aInstance, const otIp6Address *aIp6Addresses, uint8_t aNumAddresses)

Sets/updates the list of host IPv6 address.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aIp6Addresses	A pointer to the an array containing the host IPv6 addresses.
[in]	aNumAddresses	The number of addresses in the `aIp6Addresses` array.

Host IPv6 addresses can be set/changed before start or during operation of SRP client (e.g. to add/remove or change a previously registered host address), except when the host info is being removed (client is busy handling a remove request from an earlier call to `otSrpClientRemoveHostAndServices()` and host info still being in either `STATE_TO_REMOVE` or `STATE_REMOVING` states).

The host IPv6 address array pointed to by `aIp6Addresses` MUST persist and remain unchanged after returning from this function (with `OT_ERROR_NONE`). OpenThread will save the pointer to the array.

After a successful call to this function, `otSrpClientCallback` will be called to report the status of the address registration with SRP server.

Calling this function disables auto host address mode if it was previously enabled from a successful call to `otSrpClientEnableAutoHostAddress()`.

Definition at line `508` of file `include/openthread/srp_client.h`

otSrpClientAddService

> otError otSrpClientAddService (otInstance *aInstance, otSrpClientService *aService)

Adds a service to be registered with server.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aService	A pointer to a `otSrpClientService` instance to add.

After a successful call to this function, `otSrpClientCallback` will be called to report the status of the service addition/registration with SRP server.

The `otSrpClientService` instance being pointed to by `aService` MUST persist and remain unchanged after returning from this function (with `OT_ERROR_NONE`). OpenThread will save the pointer to the service instance.

The `otSrpClientService` instance is not longer tracked by OpenThread and can be reclaimed only when

- It is removed explicitly by a call to `otSrpClientRemoveService()` or removed along with other services by a call to `otSrpClientRemoveHostAndServices()` and only after the otSrpClientCallback` is called indicating the service was removed. Or,
- A call to `otSrpClientClearHostAndServices()` which removes the host and all related services immediately.

Definition at line `535` of file `include/openthread/srp_client.h`

otSrpClientRemoveService

> otError otSrpClientRemoveService (otInstance *aInstance, otSrpClientService *aService)

Requests a service to be unregistered with server.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aService	A pointer to a `otSrpClientService` instance to remove.

After a successful call to this function, `otSrpClientCallback` will be called to report the status of remove request with SRP server.

The `otSrpClientService` instance being pointed to by `aService` MUST persist and remain unchanged after returning from this function (with `OT_ERROR_NONE`). OpenThread will keep the service instance during the remove process. Only after the `otSrpClientCallback` is called indicating the service instance is removed from SRP client service list and can be be freed/reused.

Definition at line `556` of file `include/openthread/srp_client.h`

otSrpClientClearService

> otError otSrpClientClearService (otInstance *aInstance, otSrpClientService *aService)

Clears a service, immediately removing it from the client service list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aService	A pointer to a `otSrpClientService` instance to delete.

Unlike `otSrpClientRemoveService()` which sends an update message to the server to remove the service, this function clears the service from the client's service list without any interaction with the server. On a successful call to this function, the `otSrpClientCallback` will NOT be called and the `aService` entry can be reclaimed and re-used by the caller immediately.

Can be used along with a subsequent call to `otSrpClientAddService()` (potentially reusing the same `aService` entry with the same service and instance names) to update some of the parameters in an existing service.

Definition at line `576` of file `include/openthread/srp_client.h`

otSrpClientGetServices

```
const otSrpClientService * otSrpClientGetServices (otInstance *aInstance)
```

Gets the list of services being managed by client.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Returns

- A pointer to the head of linked-list of all services or NULL if the list is empty.

Definition at line `586` of file `include/openthread/srp_client.h`

otSrpClientRemoveHostAndServices

```
otError otSrpClientRemoveHostAndServices (otInstance *aInstance, bool aRemoveKeyLease, bool aSendUnregToServer)
```

Starts the remove process of the host info and all services.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aRemoveKeyLease	A boolean indicating whether or not the host key lease should also be removed.
[in]	aSendUnregToServer	A boolean indicating whether to send update to server when host info is not registered.

After returning from this function, `otSrpClientCallback` will be called to report the status of remove request with SRP server.

If the host info is to be permanently removed from server, `aRemoveKeyLease` should be set to `true` which removes the key lease associated with host on server. Otherwise, the key lease record is kept as before, which ensures that the server holds the host name in reserve for when the client is once again able to provide and register its service(s).

The `aSendUnregToServer` determines the behavior when the host info is not yet registered with the server. If `aSendUnregToServer` is set to `false` (which is the default/expected value) then the SRP client will immediately remove the host info and services without sending an update message to server (no need to update the server if nothing is yet registered with it). If `aSendUnregToServer` is set to `true` then the SRP client will send an update message to the server. Note that if the host info is registered then the value of `aSendUnregToServer` does not matter and the SRP client will always send an update message to server requesting removal of all info.

One situation where `aSendUnregToServer` can be useful is on a device reset/reboot, caller may want to remove any previously registered services with the server. In this case, caller can `otSrpClientSetHostName()` and then request `otSrpClientRemoveHostAndServices()` with `aSendUnregToServer` as `true`.

Definition at line `619` of file `include/openthread/srp_client.h`

otSrpClientClearHostAndServices

```
void otSrpClientClearHostAndServices (otInstance *aInstance)
```

Clears all host info and all the services.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Unlike `otSrpClientRemoveHostAndServices()` which sends an update message to the server to remove all the info, this function clears all the info immediately without any interaction with the server.

Definition at line `630` of file `include/openthread/srp_client.h`

otSrpClientGetDomainName

```
const char * otSrpClientGetDomainName (otInstance *aInstance)
```

Gets the domain name being used by SRP client.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Requires `OPENTHREAD_CONFIG_SRP_CLIENT_DOMAIN_NAME_API_ENABLE` to be enabled.

If domain name is not set, "default.service.arpa" will be used.

Returns

- The domain name string.

Definition at line `644` of file `include/openthread/srp_client.h`

otSrpClientSetDomainName

```
otError otSrpClientSetDomainName (otInstance *aInstance, const char *aName)
```

Sets the domain name to be used by SRP client.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aName	A pointer to the domain name string. If NULL sets it to default "default.service.arpa".

Requires `OPENTHREAD_CONFIG_SRP_CLIENT_DOMAIN_NAME_API_ENABLE` to be enabled.

If not set "default.service.arpa" will be used.

The name string buffer pointed to by `aName` MUST persist and stay unchanged after returning from this function. OpenThread will keep the pointer to the string.

The domain name can be set before client is started or after start but before host info is registered with server (host info should be in either `STATE_TO_ADD` or `STATE_TO_REMOVE`).

Definition at line `666` of file `include/openthread/srp_client.h`

otSrpClientItemStateToString

```
const char * otSrpClientItemStateToString (otSrpClientItemState aItemState)
```

Converts a `otSrpClientItemState` to a string.

Parameters

[in]	aItemState	An item state.

Returns

- A string representation of `aItemState` .

Definition at line 676 of file include/openthread/srp_client.h

otSrpClientSetServiceKeyRecordEnabled

```
void otSrpClientSetServiceKeyRecordEnabled (otInstance *aInstance, bool aEnabled)
```

Enables/disables "service key record inclusion" mode.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aEnabled	TRUE to enable, FALSE to disable the "service key record inclusion" mode.

When enabled, SRP client will include KEY record in Service Description Instructions in the SRP update messages that it sends.

Is available when OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE configuration is enabled.

Note

- KEY record is optional in Service Description Instruction (it is required and always included in the Host Description Instruction). The default behavior of SRP client is to not include it. This function is intended to override the default behavior for testing only.

Definition at line 694 of file include/openthread/srp_client.h

otSrpClientIsServiceKeyRecordEnabled

```
bool otSrpClientIsServiceKeyRecordEnabled (otInstance *aInstance)
```

Indicates whether the "service key record inclusion" mode is enabled or disabled.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Is available when OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE configuration is enabled.

Returns

- TRUE if "service key record inclusion" mode is enabled, FALSE otherwise.

Definition at line 706 of file include/openthread/srp_client.h

otSrpClientBuffersGetHostNameString

```
char * otSrpClientBuffersGetHostNameString (otInstance *aInstance, uint16_t *aSize)
```

Gets the string buffer to use for SRP client host name.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[out]	aSize	Pointer to a variable to return the size (number of bytes) of the string buffer (MUST NOT be NULL).

Returns

- A pointer to char buffer to use for SRP client host name.

Definition at line 77 of file include/openthread/srp_client_buffers.h

otSrpClientBuffersGetHostAddressesArray

```
otIp6Address * otSrpClientBuffersGetHostAddressesArray (otInstance *aInstance, uint8_t *aArrayLength)
```

Gets the array of IPv6 address entries to use as SRP client host address list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[out]	aArrayLength	Pointer to a variable to return the array length i.e., number of IPv6 address entries in the array (MUST NOT be NULL).

Returns

- A pointer to an array of otIp6Address entries (number of entries is returned in aArrayLength).

Definition at line 89 of file include/openthread/srp_client_buffers.h

otSrpClientBuffersAllocateService

```
otSrpClientBuffersServiceEntry * otSrpClientBuffersAllocateService (otInstance *aInstance)
```

Allocates a new service entry from the pool.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

The returned service entry instance will be initialized as follows:

- mService.mName will point to an allocated string buffer which can be retrieved using the function otSrpClientBuffersGetServiceEntryServiceNameString().
- mService.mInstanceName will point to an allocated string buffer which can be retrieved using the function otSrpClientBuffersGetServiceEntryInstanceNameString().
- mService.mSubTypeLabels points to an array that is returned from otSrpClientBuffersGetSubTypeLabelsArray().
- mService.mTxtEntries will point to mTxtEntry.
- mService.mNumTxtEntries will be set to one.
- Other mService fields (port, priority, weight) are set to zero.
- mTxtEntry.mKey is set to NULL (value is treated as already encoded).
- mTxtEntry.mValue will point to an allocated buffer which can be retrieved using the function otSrpClientBuffersGetServiceEntryTxtBuffer().
- mTxtEntry.mValueLength is set to zero.
- All related data/string buffers and arrays are cleared to all zero.

Returns

- A pointer to the newly allocated service entry or NULL if not more entry available in the pool.

Definition at line 115 of file include/openthread/srp_client_buffers.h

otSrpClientBuffersFreeService

```
void otSrpClientBuffersFreeService (otInstance *aInstance, otSrpClientBuffersServiceEntry *aService)
```

Frees a previously allocated service entry.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aService	A pointer to the service entry to free (MUST NOT be NULL).

The `aService` MUST be previously allocated using `otSrpClientBuffersAllocateService()` and not yet freed. Otherwise the behavior of this function is undefined.

Definition at line `127` of file `include/openthread/srp_client_buffers.h`

otSrpClientBuffersFreeAllServices

```
void otSrpClientBuffersFreeAllServices (otInstance *aInstance)
```

Frees all previously allocated service entries.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Definition at line `135` of file `include/openthread/srp_client_buffers.h`

otSrpClientBuffersGetServiceEntryServiceNameString

```
char * otSrpClientBuffersGetServiceEntryServiceNameString (otSrpClientBuffersServiceEntry *aEntry, uint16_t *aSize)
```

Gets the string buffer for service name from a service entry.

Parameters

[in]	aEntry	A pointer to a previously allocated service entry (MUST NOT be NULL).
[out]	aSize	A pointer to a variable to return the size (number of bytes) of the string buffer (MUST NOT be NULL).

Returns

- A pointer to the string buffer.

Definition at line `147` of file `include/openthread/srp_client_buffers.h`

otSrpClientBuffersGetServiceEntryInstanceNameString

```
char * otSrpClientBuffersGetServiceEntryInstanceNameString (otSrpClientBuffersServiceEntry *aEntry, uint16_t *aSize)
```

Gets the string buffer for service instance name from a service entry.

Parameters

[in]	aEntry	A pointer to a previously allocated service entry (MUST NOT be NULL).
[out]	aSize	A pointer to a variable to return the size (number of bytes) of the string buffer (MUST NOT be NULL).

Returns

- A pointer to the string buffer.

Definition at line `159` of file `include/openthread/srp_client_buffers.h`

otSrpClientBuffersGetServiceEntryTxtBuffer

```
uint8_t * otSrpClientBuffersGetServiceEntryTxtBuffer (otSrpClientBuffersServiceEntry *aEntry, uint16_t *aSize)
```

Gets the buffer for TXT record from a service entry.

Parameters

[in]	aEntry	A pointer to a previously allocated service entry (MUST NOT be NULL).
[out]	aSize	A pointer to a variable to return the size (number of bytes) of the buffer (MUST NOT be NULL).

Returns

- A pointer to the buffer.

Definition at line `170` of file `include/openthread/srp_client_buffers.h`

otSrpClientBuffersGetSubTypeLabelsArray

```
const char ** otSrpClientBuffersGetSubTypeLabelsArray (otSrpClientBuffersServiceEntry *aEntry, uint16_t *aArrayLength)
```

Gets the array for service subtype labels from the service entry.

Parameters

[in]	aEntry	A pointer to a previously allocated service entry (MUST NOT be NULL).
[out]	aArrayLength	A pointer to a variable to return the array length (MUST NOT be NULL).

Returns

- A pointer to the array.

Definition at line `181` of file `include/openthread/srp_client_buffers.h`

otSrpServerGetDomain

```
const char * otSrpServerGetDomain (otInstance *aInstance)
```

Returns the domain authorized to the SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

If the domain if not set by SetDomain, "default.service.arpa." will be returned. A trailing dot is always appended even if the domain is set without it.

Returns

- A pointer to the dot-joined domain string.

Definition at line `159` of file `include/openthread/srp_server.h`

otSrpServerSetDomain

```
otError otSrpServerSetDomain (otInstance *aInstance, const char *aDomain)
```

Sets the domain on the SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDomain	The domain to be set. MUST NOT be NULL.

A trailing dot will be appended to `aDomain` if it is not already there. Should only be called before the SRP server is enabled.

Definition at line `176` of file `include/openthread/srp_server.h`

otSrpServerGetState

```
otSrpServerState otSrpServerGetState (otInstance *aInstance)
```

Returns the state of the SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The current state of the SRP server.

Definition at line `186` of file `include/openthread/srp_server.h`

otSrpServerGetPort

```
uint16_t otSrpServerGetPort (otInstance *aInstance)
```

Returns the port the SRP server is listening to.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The port of the SRP server. It returns 0 if the server is not running.

Definition at line `196` of file `include/openthread/srp_server.h`

otSrpServerGetAddressMode

```
otSrpServerAddressMode otSrpServerGetAddressMode (otInstance *aInstance)
```

Returns the address mode being used by the SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The SRP server's address mode.

Definition at line 206 of file include/openthread/srp_server.h

otSrpServerSetAddressMode

> otError otSrpServerSetAddressMode (otInstance *aInstance, otSrpServerAddressMode aMode)

Sets the address mode to be used by the SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMode	The address mode to use.

Definition at line 218 of file include/openthread/srp_server.h

otSrpServerGetAnycastModeSequenceNumber

> uint8_t otSrpServerGetAnycastModeSequenceNumber (otInstance *aInstance)

Returns the sequence number used with anycast address mode.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The sequence number is included in "DNS/SRP Service Anycast Address" entry published in the Network Data.

Returns

- The anycast sequence number.

Definition at line 230 of file include/openthread/srp_server.h

otSrpServerSetAnycastModeSequenceNumber

> otError otSrpServerSetAnycastModeSequenceNumber (otInstance *aInstance, uint8_t aSequenceNumber)

Sets the sequence number used with anycast address mode.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSequenceNumber	The sequence number to use.

Definition at line 242 of file include/openthread/srp_server.h

otSrpServerSetEnabled

> void otSrpServerSetEnabled (otInstance *aInstance, bool aEnabled)

Enables/disables the SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[in]	aEnabled	A boolean to enable/disable the SRP server.

On a Border Router, it is recommended to use `otSrpServerSetAutoEnableMode()` instead.

Definition at line `253` of file `include/openthread/srp_server.h`

otSrpServerSetAutoEnableMode

> void otSrpServerSetAutoEnableMode (otInstance *aInstance, bool aEnabled)

Enables/disables the auto-enable mode on SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	A boolean to enable/disable the auto-enable mode.

Requires `OPENTHREAD_CONFIG_BORDER_ROUTING_ENABLE` feature.

When this mode is enabled, the Border Routing Manager controls if/when to enable or disable the SRP server. SRP sever is auto-enabled if/when Border Routing is started and it is done with the initial prefix and route configurations (when the OMR and on-link prefixes are determined, advertised in emitted Router Advertisement message on infrastructure side and published in the Thread Network Data). The SRP server is auto-disabled if/when BR is stopped (e.g., if the infrastructure network interface is brought down or if BR gets detached).

This mode can be disabled by a `otSrpServerSetAutoEnableMode()` call with `aEnabled` set to `false` or if the SRP server is explicitly enabled or disabled by a call to `otSrpServerSetEnabled()` function. Disabling auto-enable mode using `otSrpServerSetAutoEnableMode(false)` will not change the current state of SRP sever (e.g., if it is enabled it stays enabled).

Definition at line `275` of file `include/openthread/srp_server.h`

otSrpServerIsAutoEnableMode

> bool otSrpServerIsAutoEnableMode (otInstance *aInstance)

Indicates whether the auto-enable mode is enabled or disabled.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Requires `OPENTHREAD_CONFIG_BORDER_ROUTING_ENABLE` feature.

Definition at line `288` of file `include/openthread/srp_server.h`

otSrpServerGetTtlConfig

> void otSrpServerGetTtlConfig (otInstance *aInstance, otSrpServerTtlConfig *aTtlConfig)

Returns SRP server TTL configuration.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aTtlConfig	A pointer to an `otSrpServerTtlConfig` instance.

Definition at line 297 of file include/openthread/srp_server.h

otSrpServerSetTtlConfig

> otError otSrpServerSetTtlConfig (otInstance *aInstance, const otSrpServerTtlConfig *aTtlConfig)

Sets SRP server TTL configuration.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aTtlConfig	A pointer to an otSrpServerTtlConfig instance.

The granted TTL will always be no greater than the max lease interval configured via otSrpServerSetLeaseConfig(), regardless of the minimum and maximum TTL configuration.

Definition at line 312 of file include/openthread/srp_server.h

otSrpServerGetLeaseConfig

> void otSrpServerGetLeaseConfig (otInstance *aInstance, otSrpServerLeaseConfig *aLeaseConfig)

Returns SRP server LEASE and KEY-LEASE configurations.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aLeaseConfig	A pointer to an otSrpServerLeaseConfig instance.

Definition at line 321 of file include/openthread/srp_server.h

otSrpServerSetLeaseConfig

> otError otSrpServerSetLeaseConfig (otInstance *aInstance, const otSrpServerLeaseConfig *aLeaseConfig)

Sets SRP server LEASE and KEY-LEASE configurations.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aLeaseConfig	A pointer to an otSrpServerLeaseConfig instance.

When a non-zero LEASE time is requested from a client, the granted value will be limited in range [aMinLease, aMaxLease]; and a non-zero KEY-LEASE will be granted in range [aMinKeyLease, aMaxKeyLease]. For zero LEASE or KEY-LEASE time, zero will be granted.

Definition at line 338 of file include/openthread/srp_server.h

otSrpServerSetServiceUpdateHandler

> void otSrpServerSetServiceUpdateHandler (otInstance *aInstance, otSrpServerServiceUpdateHandler aServiceHandler, void *aContext)

Sets the SRP service updates handler on SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aServiceHandler	A pointer to a service handler. Use NULL to remove the handler.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.

Definition at line 387 of file include/openthread/srp_server.h

otSrpServerHandleServiceUpdateResult

```
void otSrpServerHandleServiceUpdateResult (otInstance *aInstance, otSrpServerServiceUpdateId aId, otError aError)
```

Reports the result of processing a SRP update to the SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aId	The service update transaction ID. This should be the same ID provided via otSrpServerServiceUpdateHandler .
[in]	aError	An error to be returned to the SRP server. Use OT_ERROR_DUPLICATED to represent DNS name conflicts.

The Service Update Handler should call this function to return the result of its processing of a SRP update.

Definition at line 404 of file include/openthread/srp_server.h

otSrpServerGetNextHost

```
const otSrpServerHost * otSrpServerGetNextHost (otInstance *aInstance, const otSrpServerHost *aHost)
```

Returns the next registered host on the SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aHost	A pointer to current host; use NULL to get the first host.

Returns

- A pointer to the registered host. NULL, if no more hosts can be found.

Definition at line 415 of file include/openthread/srp_server.h

otSrpServerGetResponseCounters

```
const otSrpServerResponseCounters * otSrpServerGetResponseCounters (otInstance *aInstance)
```

Returns the response counters of the SRP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the response counters of the SRP server.

Definition at line 425 of file include/openthread/srp_server.h

otSrpServerHostIsDeleted

```
bool otSrpServerHostIsDeleted (const otSrpServerHost *aHost)
```

Tells if the SRP service host has been deleted.

Parameters

[in]	aHost	A pointer to the SRP service host.

A SRP service host can be deleted but retains its name for future uses. In this case, the host instance is not removed from the SRP server/registry.

Returns

- TRUE if the host has been deleted, FALSE if not.

Definition at line 438 of file include/openthread/srp_server.h

otSrpServerHostGetFullName

```
const char * otSrpServerHostGetFullName (const otSrpServerHost *aHost)
```

Returns the full name of the host.

Parameters

[in]	aHost	A pointer to the SRP service host.

Returns

- A pointer to the null-terminated host name string.

Definition at line 448 of file include/openthread/srp_server.h

otSrpServerHostMatchesFullName

```
bool otSrpServerHostMatchesFullName (const otSrpServerHost *aHost, const char *aFullName)
```

Indicates whether the host matches a given host name.

Parameters

[in]	aHost	A pointer to the SRP service host.
[in]	aFullName	A full host name.

DNS name matches are performed using a case-insensitive string comparison (i.e., "Abc" and "aBc" are considered to be the same).

Definition at line 463 of file include/openthread/srp_server.h

otSrpServerHostGetAddresses

```
const otIp6Address * otSrpServerHostGetAddresses (const otSrpServerHost *aHost, uint8_t *aAddressesNum)
```

Returns the addresses of given host.

Parameters

[in]	aHost	A pointer to the SRP service host.
[out]	aAddressesNum	A pointer to where we should output the number of the addresses to.

Returns

- A pointer to the array of IPv6 Address.

Definition at line `474` of file `include/openthread/srp_server.h`

otSrpServerHostGetLeaseInfo

```
void otSrpServerHostGetLeaseInfo (const otSrpServerHost *aHost, otSrpServerLeaseInfo *aLeaseInfo)
```

Returns the LEASE and KEY-LEASE information of a given host.

Parameters

[in]	aHost	A pointer to the SRP server host.
[out]	aLeaseInfo	A pointer to where to output the LEASE and KEY-LEASE information.

Definition at line `483` of file `include/openthread/srp_server.h`

otSrpServerHostGetNextService

```
const otSrpServerService * otSrpServerHostGetNextService (const otSrpServerHost *aHost, const otSrpServerService *aService)
```

Returns the next service of given host.

Parameters

[in]	aHost	A pointer to the SRP service host.
[in]	aService	A pointer to current SRP service instance; use NULL to get the first service.

Returns

- A pointer to the next service or NULL if there is no more services.

Definition at line `494` of file `include/openthread/srp_server.h`

otSrpServerServiceIsDeleted

```
bool otSrpServerServiceIsDeleted (const otSrpServerService *aService)
```

Indicates whether or not the SRP service has been deleted.

Parameters

[in]	aService	A pointer to the SRP service.

A SRP service can be deleted but retains its name for future uses. In this case, the service instance is not removed from the SRP server/registry. It is guaranteed that all services are deleted if the host is deleted.

Returns

- TRUE if the service has been deleted, FALSE if not.

Definition at line 509 of file include/openthread/srp_server.h

otSrpServerServiceGetInstanceName

```
const char * otSrpServerServiceGetInstanceName (const otSrpServerService *aService)
```

Returns the full service instance name of the service.

Parameters

[in]	aService	A pointer to the SRP service.

Returns

- A pointer to the null-terminated service instance name string.

Definition at line 519 of file include/openthread/srp_server.h

otSrpServerServiceMatchesInstanceName

```
bool otSrpServerServiceMatchesInstanceName (const otSrpServerService *aService, const char *aInstanceName)
```

Indicates whether this service matches a given service instance name.

Parameters

[in]	aService	A pointer to the SRP service.
[in]	aInstanceName	The service instance name.

DNS name matches are performed using a case-insensitive string comparison (i.e., "Abc" and "aBc" are considered to be the same).

Definition at line 534 of file include/openthread/srp_server.h

otSrpServerServiceGetInstanceLabel

```
const char * otSrpServerServiceGetInstanceLabel (const otSrpServerService *aService)
```

Returns the service instance label (first label in instance name) of the service.

Parameters

[in]	aService	A pointer to the SRP service.

Returns

- A pointer to the null-terminated service instance label string..

Definition at line 544 of file include/openthread/srp_server.h

otSrpServerServiceGetServiceName

const char * otSrpServerServiceGetServiceName (const otSrpServerService *aService)

Returns the full service name of the service.

Parameters

[in]	aService	A pointer to the SRP service.

Returns

- A pointer to the null-terminated service name string.

Definition at line `554` of file `include/openthread/srp_server.h`

otSrpServerServiceMatchesServiceName

bool otSrpServerServiceMatchesServiceName (const otSrpServerService *aService, const char *aServiceName)

Indicates whether this service matches a given service name.

Parameters

[in]	aService	A pointer to the SRP service.
[in]	aServiceName	The service name.

DNS name matches are performed using a case-insensitive string comparison (i.e., "Abc" and "aBc" are considered to be the same).

Definition at line `569` of file `include/openthread/srp_server.h`

otSrpServerServiceGetNumberOfSubTypes

uint16_t otSrpServerServiceGetNumberOfSubTypes (const otSrpServerService *aService)

Gets the number of sub-types of the service.

Parameters

[in]	aService	A pointer to the SRP service.

Returns

- The number of sub-types of `aService` .

Definition at line `579` of file `include/openthread/srp_server.h`

otSrpServerServiceGetSubTypeServiceNameAt

const char * otSrpServerServiceGetSubTypeServiceNameAt (const otSrpServerService *aService, uint16_t aIndex)

Gets the sub-type service name (full name) of the service at a given index.

Parameters

[in]	aService	A pointer to the SRP service.
[in]	aIndex	The index to get.

The full service name for a sub-type service follows "<sub-label>._sub.<service-labels>.<domain>.".

Returns

- A pointer to sub-type service name at `aIndex`, or `NULL` if no sub-type at this index.

Definition at line `592` of file `include/openthread/srp_server.h`

otSrpServerServiceHasSubTypeServiceName

```
bool otSrpServerServiceHasSubTypeServiceName (const otSrpServerService *aService, const char *aSubTypeServiceName)
```

Indicates whether or not the service has a given sub-type.

Parameters

[in]	aService	A pointer to the SRP service.
[in]	aSubTypeServiceName	The sub-type service name (full name) to check.

DNS name matches are performed using a case-insensitive string comparison (i.e., "Abc" and "aBc" are considered to be the same).

Definition at line `607` of file `include/openthread/srp_server.h`

otSrpServerParseSubTypeServiceName

```
otError otSrpServerParseSubTypeServiceName (const char *aSubTypeServiceName, char *aLabel, uint8_t aLabelSize)
```

Parses a sub-type service name (full name) and extracts the sub-type label.

Parameters

[in]	aSubTypeServiceName	A sub-type service name (full name).
[out]	aLabel	A pointer to a buffer to copy the extracted sub-type label.
[in]	aLabelSize	Maximum size of `aLabel` buffer.

The full service name for a sub-type service follows "<sub-label>._sub.<service-labels>.<domain>.".

Definition at line `624` of file `include/openthread/srp_server.h`

otSrpServerServiceGetPort

```
uint16_t otSrpServerServiceGetPort (const otSrpServerService *aService)
```

Returns the port of the service instance.

Parameters

[in]	aService	A pointer to the SRP service.

Returns

- The port of the service.

Definition at line `634` of file `include/openthread/srp_server.h`

otSrpServerServiceGetWeight

```
uint16_t otSrpServerServiceGetWeight (const otSrpServerService *aService)
```

Returns the weight of the service instance.

Parameters

[in]	aService	A pointer to the SRP service.

Returns

- The weight of the service.

Definition at line `644` of file `include/openthread/srp_server.h`

otSrpServerServiceGetPriority

```
uint16_t otSrpServerServiceGetPriority (const otSrpServerService *aService)
```

Returns the priority of the service instance.

Parameters

[in]	aService	A pointer to the SRP service.

Returns

- The priority of the service.

Definition at line `654` of file `include/openthread/srp_server.h`

otSrpServerServiceGetTtl

```
uint32_t otSrpServerServiceGetTtl (const otSrpServerService *aService)
```

Returns the TTL of the service instance.

Parameters

[in]	aService	A pointer to the SRP service.

Returns

- The TTL of the service instance..

Definition at line `664` of file `include/openthread/srp_server.h`

otSrpServerServiceGetTxtData

```
const uint8_t * otSrpServerServiceGetTxtData (const otSrpServerService *aService, uint16_t *aDataLength)
```

Returns the TXT record data of the service instance.

Parameters

[in]	aService	A pointer to the SRP service.

[out]	aDataLength	A pointer to return the TXT record data length. MUST NOT be NULL.

Returns

- A pointer to the buffer containing the TXT record data (the TXT data length is returned in `aDataLength`).

Definition at line `675` of file `include/openthread/srp_server.h`

otSrpServerServiceGetHost

```
const otSrpServerHost * otSrpServerServiceGetHost (const otSrpServerService *aService)
```

Returns the host which the service instance reside on.

Parameters

[in]	aService	A pointer to the SRP service.

Returns

- A pointer to the host instance.

Definition at line `685` of file `include/openthread/srp_server.h`

otSrpServerServiceGetLeaseInfo

```
void otSrpServerServiceGetLeaseInfo (const otSrpServerService *aService, otSrpServerLeaseInfo *aLeaseInfo)
```

Returns the LEASE and KEY-LEASE information of a given service.

Parameters

[in]	aService	A pointer to the SRP server service.
[out]	aLeaseInfo	A pointer to where to output the LEASE and KEY-LEASE information.

Definition at line `694` of file `include/openthread/srp_server.h`

otSrpClientHostInfo

Represents an SRP client host info.

Public Attributes

const char *	mName Host name (label) string (NULL if not yet set).
const otIp6Address *	mAddresses Array of host IPv6 addresses (NULL if not set or auto address is enabled).
uint8_t	mNumAddresses Number of IPv6 addresses in `mAddresses` array.
bool	mAutoAddress Indicates whether auto address mode is enabled or not.
otSrpClientItemState	mState Host info state.

Public Attribute Documentation

mName

```
const char* otSrpClientHostInfo::mName
```

Host name (label) string (NULL if not yet set).

Definition at line `77` of file `include/openthread/srp_client.h`

mAddresses

```
const otIp6Address* otSrpClientHostInfo::mAddresses
```

Array of host IPv6 addresses (NULL if not set or auto address is enabled).

Definition at line `78` of file `include/openthread/srp_client.h`

mNumAddresses

```
uint8_t otSrpClientHostInfo::mNumAddresses
```

Number of IPv6 addresses in `mAddresses` array.

Definition at line `79` of file `include/openthread/srp_client.h`

mAutoAddress

```
bool otSrpClientHostInfo::mAutoAddress
```

Indicates whether auto address mode is enabled or not.

Definition at line 80 of file include/openthread/srp_client.h

mState

```
otSrpClientItemState otSrpClientHostInfo::mState
```

Host info state.

Definition at line 81 of file include/openthread/srp_client.h

otSrpClientService

Represents an SRP client service.

The values in this structure, including the string buffers for the names and the TXT record entries, MUST persist and stay constant after an instance of this structure is passed to OpenThread from `otSrpClientAddService()` or `otSrpClientRemoveService()`.

The `mState`, `mData`, `mNext` fields are used/managed by OT core only. Their value is ignored when an instance of `otSrpClientService` is passed in `otSrpClientAddService()` or `otSrpClientRemoveService()` or other functions. The caller does not need to set these fields.

The `mLease` and `mKeyLease` fields specify the desired lease and key lease intervals for this service. Zero value indicates that the interval is unspecified and then the default lease or key lease intervals from `otSrpClientGetLeaseInterval()` and `otSrpClientGetKeyLeaseInterval()` are used for this service. If the key lease interval (whether set explicitly or determined from the default) is shorter than the lease interval for a service, SRP client will re-use the lease interval value for key lease interval as well. For example, if in service `mLease` is explicitly set to 2 days and `mKeyLease` is set to zero and default key lease is set to 1 day, then when registering this service, the requested key lease for this service is also set to 2 days.

Public Attributes

const char *	mName The service labels (e.g., "_mt._udp", not the full domain name).
const char *	mInstanceName The service instance name label (not the full name).
const char *const *	mSubTypeLabels Array of sub-type labels (must end with `NULL` or can be `NULL`).
const otDnsTxtEntry *	mTxtEntries Array of TXT entries (`mNumTxtEntries` gives num of entries).
uint16_t	mPort The service port number.
uint16_t	mPriority The service priority.
uint16_t	mWeight The service weight.
uint8_t	mNumTxtEntries Number of entries in the `mTxtEntries` array.
otSrpClientItemState	mState Service state (managed by OT core).
uint32_t	mData Internal data (used by OT core).
struct otSrpClientService *	mNext Pointer to next entry in a linked-list (managed by OT core).

uint32_t mLease
Desired lease interval in sec - zero to use default.

uint32_t mKeyLease
Desired key lease interval in sec - zero to use default.

Public Attribute Documentation

mName

```
const char* otSrpClientService::mName
```

The service labels (e.g., "_mt._udp", not the full domain name).

Definition at line 106 of file include/openthread/srp_client.h

mInstanceName

```
const char* otSrpClientService::mInstanceName
```

The service instance name label (not the full name).

Definition at line 107 of file include/openthread/srp_client.h

mSubTypeLabels

```
const char* const* otSrpClientService::mSubTypeLabels
```

Array of sub-type labels (must end with NULL or can be NULL).

Definition at line 108 of file include/openthread/srp_client.h

mTxtEntries

```
const otDnsTxtEntry* otSrpClientService::mTxtEntries
```

Array of TXT entries (mNumTxtEntries gives num of entries).

Definition at line 109 of file include/openthread/srp_client.h

mPort

```
uint16_t otSrpClientService::mPort
```

The service port number.

Definition at line 110 of file include/openthread/srp_client.h

mPriority

uint16_t otSrpClientService::mPriority

The service priority.

Definition at line 111 of file include/openthread/srp_client.h

mWeight

uint16_t otSrpClientService::mWeight

The service weight.

Definition at line 112 of file include/openthread/srp_client.h

mNumTxtEntries

uint8_t otSrpClientService::mNumTxtEntries

Number of entries in the mTxtEntries array.

Definition at line 113 of file include/openthread/srp_client.h

mState

otSrpClientItemState otSrpClientService::mState

Service state (managed by OT core).

Definition at line 114 of file include/openthread/srp_client.h

mData

uint32_t otSrpClientService::mData

Internal data (used by OT core).

Definition at line 115 of file include/openthread/srp_client.h

mNext

struct otSrpClientService* otSrpClientService::mNext

Pointer to next entry in a linked-list (managed by OT core).

Definition at line 116 of file include/openthread/srp_client.h

mLease

uint32_t otSrpClientService::mLease

Desired lease interval in sec - zero to use default.

Definition at line 117 of file include/openthread/srp_client.h

mKeyLease

uint32_t otSrpClientService::mKeyLease

Desired key lease interval in sec - zero to use default.

Definition at line 118 of file include/openthread/srp_client.h

otSrpClientBuffersServiceEntry

Represents a SRP client service pool entry.

Public Attributes

otSrpClientService	mService The SRP client service structure.
otDnsTxtEntry	mTxtEntry The SRP client TXT entry.

Public Attribute Documentation

mService

```
otSrpClientService otSrpClientBuffersServiceEntry::mService
```

The SRP client service structure.

Definition at line 63 of file include/openthread/srp_client_buffers.h

mTxtEntry

```
otDnsTxtEntry otSrpClientBuffersServiceEntry::mTxtEntry
```

The SRP client TXT entry.

Definition at line 64 of file include/openthread/srp_client_buffers.h

otSrpServerTtlConfig

Includes SRP server TTL configurations.

Public Attributes

uint32_t	mMinTtl The minimum TTL in seconds.
uint32_t	mMaxTtl The maximum TTL in seconds.

Public Attribute Documentation

mMinTtl

```
uint32_t otSrpServerTtlConfig::mMinTtl
```

The minimum TTL in seconds.

Definition at line 106 of file include/openthread/srp_server.h

mMaxTtl

```
uint32_t otSrpServerTtlConfig::mMaxTtl
```

The maximum TTL in seconds.

Definition at line 107 of file include/openthread/srp_server.h

otSrpServerLeaseConfig

Includes SRP server LEASE and KEY-LEASE configurations.

Public Attributes

uint32_t	mMinLease The minimum LEASE interval in seconds.
uint32_t	mMaxLease The maximum LEASE interval in seconds.
uint32_t	mMinKeyLease The minimum KEY-LEASE interval in seconds.
uint32_t	mMaxKeyLease The maximum KEY-LEASE interval in seconds.

Public Attribute Documentation

mMinLease

```
uint32_t otSrpServerLeaseConfig::mMinLease
```

The minimum LEASE interval in seconds.

Definition at line 116 of file include/openthread/srp_server.h

mMaxLease

```
uint32_t otSrpServerLeaseConfig::mMaxLease
```

The maximum LEASE interval in seconds.

Definition at line 117 of file include/openthread/srp_server.h

mMinKeyLease

```
uint32_t otSrpServerLeaseConfig::mMinKeyLease
```

The minimum KEY-LEASE interval in seconds.

Definition at line 118 of file include/openthread/srp_server.h

mMaxKeyLease

```
uint32_t otSrpServerLeaseConfig::mMaxKeyLease
```

The maximum KEY-LEASE interval in seconds.

Definition at line 119 of file include/openthread/srp_server.h

otSrpServerLeaseInfo

Includes SRP server lease information of a host/service.

Public Attributes

uint32_t	mLease The lease time of a host/service in milliseconds.
uint32_t	mKeyLease The key lease time of a host/service in milliseconds.
uint32_t	mRemainingLease The remaining lease time of the host/service in milliseconds.
uint32_t	mRemainingKeyLease The remaining key lease time of a host/service in milliseconds.

Public Attribute Documentation

mLease

```
uint32_t otSrpServerLeaseInfo::mLease
```

The lease time of a host/service in milliseconds.

Definition at line 128 of file include/openthread/srp_server.h

mKeyLease

```
uint32_t otSrpServerLeaseInfo::mKeyLease
```

The key lease time of a host/service in milliseconds.

Definition at line 129 of file include/openthread/srp_server.h

mRemainingLease

```
uint32_t otSrpServerLeaseInfo::mRemainingLease
```

The remaining lease time of the host/service in milliseconds.

Definition at line 130 of file include/openthread/srp_server.h

mRemainingKeyLease

```
uint32_t otSrpServerLeaseInfo::mRemainingKeyLease
```

The remaining key lease time of a host/service in milliseconds.

Definition at line 131 of file include/openthread/srp_server.h

otSrpServerResponseCounters

Includes the statistics of SRP server responses.

Public Attributes

uint32_t	mSuccess The number of successful responses.
uint32_t	mServerFailure The number of server failure responses.
uint32_t	mFormatError The number of format error responses.
uint32_t	mNameExists The number of 'name exists' responses.
uint32_t	mRefused The number of refused responses.
uint32_t	mOther The number of other responses.

Public Attribute Documentation

mSuccess

```
uint32_t otSrpServerResponseCounters::mSuccess
```

The number of successful responses.

Definition at line 140 of file include/openthread/srp_server.h

mServerFailure

```
uint32_t otSrpServerResponseCounters::mServerFailure
```

The number of server failure responses.

Definition at line 141 of file include/openthread/srp_server.h

mFormatError

```
uint32_t otSrpServerResponseCounters::mFormatError
```

The number of format error responses.

Definition at line 142 of file include/openthread/srp_server.h

uint32_t otSrpServerResponseCounters::mNameExists

The number of 'name exists' responses.

Definition at line 143 of file include/openthread/srp_server.h

mRefused

uint32_t otSrpServerResponseCounters::mRefused

The number of refused responses.

Definition at line 144 of file include/openthread/srp_server.h

mOther

uint32_t otSrpServerResponseCounters::mOther

The number of other responses.

Definition at line 145 of file include/openthread/srp_server.h

Ping Sender

Ping Sender

This file includes the OpenThread API for the ping sender module.

Modules

otPingSenderReply

otPingSenderStatistics

otPingSenderConfig

Typedefs

typedef struct otPingSenderReply	otPingSenderReply Represents a ping reply.
typedef struct otPingSenderStatistics	otPingSenderStatistics Represents statistics of a ping request.
typedef void(*	otPingSenderReplyCallback)(const otPingSenderReply *aReply, void *aContext) Pointer type specifies the callback to notify receipt of a ping reply.
typedef void(*	otPingSenderStatisticsCallback)(const otPingSenderStatistics *aStatistics, void *aContext) Pointer type specifies the callback to report the ping statistics.
typedef struct otPingSenderConfig	otPingSenderConfig Represents a ping request configuration.

Functions

otError	otPingSenderPing(otInstance *aInstance, const otPingSenderConfig *aConfig) Starts a ping.
void	otPingSenderStop(otInstance *aInstance) Stops an ongoing ping.

Typedef Documentation

otPingSenderReply

```
typedef struct otPingSenderReply otPingSenderReply
```

Represents a ping reply.

Definition at line 69 of file include/openthread/ping_sender.h

otPingSenderStatistics

```
typedef struct otPingSenderStatistics otPingSenderStatistics
```

Represents statistics of a ping request.

Definition at line `83` of file `include/openthread/ping_sender.h`

otPingSenderReplyCallback

```
typedef void(* otPingSenderReplyCallback) (const otPingSenderReply *aReply, void *aContext) )(const otPingSenderReply *aReply, void *aContext)
```

Pointer type specifies the callback to notify receipt of a ping reply.

Parameters

[in]	aReply	A pointer to a `otPingSenderReply` containing info about the received ping reply.
[in]	aContext	A pointer to application-specific context.

Definition at line `92` of file `include/openthread/ping_sender.h`

otPingSenderStatisticsCallback

```
typedef void(* otPingSenderStatisticsCallback) (const otPingSenderStatistics *aStatistics, void *aContext) )(const otPingSenderStatistics *aStatistics, void *aContext)
```

Pointer type specifies the callback to report the ping statistics.

Parameters

[in]	aStatistics	A pointer to a `otPingSenderStatistics` containing info about the received ping statistics.
[in]	aContext	A pointer to application-specific context.

Definition at line `102` of file `include/openthread/ping_sender.h`

otPingSenderConfig

```
typedef struct otPingSenderConfig otPingSenderConfig
```

Represents a ping request configuration.

Definition at line `124` of file `include/openthread/ping_sender.h`

Function Documentation

otPingSenderPing

```
otError otPingSenderPing (otInstance *aInstance, const otPingSenderConfig *aConfig)
```

Starts a ping.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aConfig	The ping config to use.

Definition at line 138 of file include/openthread/ping_sender.h

otPingSenderStop

```
void otPingSenderStop (otInstance *aInstance)
```

Stops an ongoing ping.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 146 of file include/openthread/ping_sender.h

otPingSenderReply

Represents a ping reply.

Public Attributes

otIp6Address	mSenderAddress Sender IPv6 address (address from which ping reply was received).
uint16_t	mRoundTripTime Round trip time in msec.
uint16_t	mSize Data size (number of bytes) in reply (excluding IPv6 and ICMP6 headers).
uint16_t	mSequenceNumber Sequence number.
uint8_t	mHopLimit Hop limit.

Public Attribute Documentation

mSenderAddress

```
otIp6Address otPingSenderReply::mSenderAddress
```

Sender IPv6 address (address from which ping reply was received).

Definition at line `64` of file `include/openthread/ping_sender.h`

mRoundTripTime

```
uint16_t otPingSenderReply::mRoundTripTime
```

Round trip time in msec.

Definition at line `65` of file `include/openthread/ping_sender.h`

mSize

```
uint16_t otPingSenderReply::mSize
```

Data size (number of bytes) in reply (excluding IPv6 and ICMP6 headers).

Definition at line `66` of file `include/openthread/ping_sender.h`

mSequenceNumber

> uint16_t otPingSenderReply::mSequenceNumber

Sequence number.

Definition at line `67` of file `include/openthread/ping_sender.h`

mHopLimit

> uint8_t otPingSenderReply::mHopLimit

Hop limit.

Definition at line `68` of file `include/openthread/ping_sender.h`

otPingSenderStatistics

Represents statistics of a ping request.

Public Attributes

uint16_t	mSentCount The number of ping requests already sent.
uint16_t	mReceivedCount The number of ping replies received.
uint32_t	mTotalRoundTripTime The total round trip time of ping requests.
uint16_t	mMinRoundTripTime The min round trip time among ping requests.
uint16_t	mMaxRoundTripTime The max round trip time among ping requests.
bool	mIsMulticast Whether this is a multicast ping request.

Public Attribute Documentation

mSentCount

```
uint16_t otPingSenderStatistics::mSentCount
```

The number of ping requests already sent.

Definition at line 77 of file include/openthread/ping_sender.h

mReceivedCount

```
uint16_t otPingSenderStatistics::mReceivedCount
```

The number of ping replies received.

Definition at line 78 of file include/openthread/ping_sender.h

mTotalRoundTripTime

```
uint32_t otPingSenderStatistics::mTotalRoundTripTime
```

The total round trip time of ping requests.

Definition at line 79 of file include/openthread/ping_sender.h

> uint16_t otPingSenderStatistics::mMinRoundTripTime

The min round trip time among ping requests.

Definition at line `80` of file `include/openthread/ping_sender.h`

mMaxRoundTripTime

> uint16_t otPingSenderStatistics::mMaxRoundTripTime

The max round trip time among ping requests.

Definition at line `81` of file `include/openthread/ping_sender.h`

mIsMulticast

> bool otPingSenderStatistics::mIsMulticast

Whether this is a multicast ping request.

Definition at line `82` of file `include/openthread/ping_sender.h`

otPingSenderConfig

Represents a ping request configuration.

Public Attributes

otIp6Address	mSource Source address of the ping.
otIp6Address	mDestination Destination address to ping.
otPingSenderReplyCallback	mReplyCallback Callback function to report replies (can be NULL if not needed).
otPingSenderStatisticsCallback	mStatisticsCallback Callback function to report statistics (can be NULL if not needed).
void *	mCallbackContext A pointer to the callback application-specific context.
uint16_t	mSize Data size (# of bytes) excludes IPv6/ICMPv6 header. Zero for default.
uint16_t	mCount Number of ping messages to send. Zero to use default.
uint32_t	mInterval Ping tx interval in milliseconds. Zero to use default.
uint16_t	mTimeout Time in milliseconds to wait for final reply after sending final request.
uint8_t	mHopLimit Hop limit (used if `mAllowZeroHopLimit` is false). Zero for default.
bool	mAllowZeroHopLimit Indicates whether hop limit is zero.
bool	mMulticastLoop Allow looping back pings to multicast address that device is subscribed to.

Public Attribute Documentation

mSource

> otIp6Address otPingSenderConfig::mSource

Source address of the ping.

Definition at line `110` of file `include/openthread/ping_sender.h`

mDestination

```
otIp6Address otPingSenderConfig::mDestination
```

Destination address to ping.

Definition at line 111 of file include/openthread/ping_sender.h

mReplyCallback

```
otPingSenderReplyCallback otPingSenderConfig::mReplyCallback
```

Callback function to report replies (can be NULL if not needed).

Definition at line 112 of file include/openthread/ping_sender.h

mStatisticsCallback

```
otPingSenderStatisticsCallback otPingSenderConfig::mStatisticsCallback
```

Callback function to report statistics (can be NULL if not needed).

Definition at line 114 of file include/openthread/ping_sender.h

mCallbackContext

```
void* otPingSenderConfig::mCallbackContext
```

A pointer to the callback application-specific context.

Definition at line 115 of file include/openthread/ping_sender.h

mSize

```
uint16_t otPingSenderConfig::mSize
```

Data size (# of bytes) excludes IPv6/ICMPv6 header. Zero for default.

Definition at line 116 of file include/openthread/ping_sender.h

mCount

```
uint16_t otPingSenderConfig::mCount
```

Number of ping messages to send. Zero to use default.

Definition at line 117 of file include/openthread/ping_sender.h

mInterval

> uint32_t otPingSenderConfig::mInterval

Ping tx interval in milliseconds. Zero to use default.

Definition at line 118 of file include/openthread/ping_sender.h

mTimeout

> uint16_t otPingSenderConfig::mTimeout

Time in milliseconds to wait for final reply after sending final request.

Zero to use default.

Definition at line 119 of file include/openthread/ping_sender.h

mHopLimit

> uint8_t otPingSenderConfig::mHopLimit

Hop limit (used if mAllowZeroHopLimit is false). Zero for default.

Definition at line 121 of file include/openthread/ping_sender.h

mAllowZeroHopLimit

> bool otPingSenderConfig::mAllowZeroHopLimit

Indicates whether hop limit is zero.

Definition at line 122 of file include/openthread/ping_sender.h

mMulticastLoop

> bool otPingSenderConfig::mMulticastLoop

Allow looping back pings to multicast address that device is subscribed to.

Definition at line 123 of file include/openthread/ping_sender.h

TCP

TCP

Modules

TCP

TCP Abstractions

TCP

TCP

This module includes functions that control TCP communication.

Modules

otLinkedBuffer

otTcpEndpoint

otTcpEndpointInitializeArgs

otTcpListener

otTcpListenerInitializeArgs

Enumerations

enum otTcpDisconnectedReason {

OT_TCP_DISCONNECTED_REASON_NORMAL
OT_TCP_DISCONNECTED_REASON_REFUSED
OT_TCP_DISCONNECTED_REASON_RESET
OT_TCP_DISCONNECTED_REASON_TIME_WAIT
OT_TCP_DISCONNECTED_REASON_TIMED_OUT

}

enum @22 {

OT_TCP_CONNECT_NO_FAST_OPEN = 1 << 0

}

Defines flags passed to otTcpConnect().

enum @23 {

OT_TCP_SEND_MORE_TO_COME = 1 << 0

}

Defines flags passed to `otTcpSendByReference`.

enum otTcpIncomingConnectionAction {

OT_TCP_INCOMING_CONNECTION_ACTION_ACCEPT
OT_TCP_INCOMING_CONNECTION_ACTION_DEFER
OT_TCP_INCOMING_CONNECTION_ACTION_REFUSE

}

Defines incoming connection actions.

Typedefs

typedef struct otLinkedBuffer otLinkedBuffer

A linked buffer structure for use with TCP.

typedef struct otTcpEndpoint otTcpEndpoint

typedef void(* otTcpEstablished)(otTcpEndpoint *aEndpoint)
This callback informs the application that the TCP 3-way handshake is complete and that the connection is now established.

typedef void(* otTcpSendDone)(otTcpEndpoint *aEndpoint, otLinkedBuffer *aData)
This callback informs the application that data in the provided `aData` have been acknowledged by the connection peer and that `aData` and the data it contains can be reclaimed by the application.

typedef void(* otTcpForwardProgress)(otTcpEndpoint *aEndpoint, size_t aInSendBuffer, size_t aBacklog)
This callback informs the application if forward progress has been made in transferring data from the send buffer to the recipient.

typedef void(* otTcpReceiveAvailable)(otTcpEndpoint *aEndpoint, size_t aBytesAvailable, bool aEndOfStream, size_t aBytesRemaining)
This callback indicates the number of bytes available for consumption from the receive buffer.

typedef enum otTcpDisconnectedReason otTcpDisconnectedReason

typedef void(* otTcpDisconnected)(otTcpEndpoint *aEndpoint, otTcpDisconnectedReason aReason)
This callback indicates that the connection was broken and should no longer be used, or that a connection has entered the TIME-WAIT state.

typedef struct otTcpEndpointInitializeArgs otTcpEndpointInitializeArgs
Contains arguments to the otTcpEndpointInitialize() function.

typedef struct otTcpListener otTcpListener

typedef enum otTcpIncomingConnectionAction otTcpIncomingConnectionAction
Defines incoming connection actions.

typedef otTcpIncomingConnectionAction(* otTcpAcceptReady)(otTcpListener *aListener, const otSockAddr *aPeer, otTcpEndpoint **aAcceptInto)
This callback indicates that an incoming connection that matches this TCP listener has arrived.

typedef void(* otTcpAcceptDone)(otTcpListener *aListener, otTcpEndpoint *aEndpoint, const otSockAddr *aPeer)
This callback indicates that the TCP connection is now ready for two-way communication.

typedef struct otTcpListenerInitializeArgs otTcpListenerInitializeArgs
Contains arguments to the otTcpListenerInitialize() function.

Functions

otError otTcpEndpointInitialize(otInstance *aInstance, otTcpEndpoint *aEndpoint, const otTcpEndpointInitializeArgs *aArgs)
Initializes a TCP endpoint.

otInstance * otTcpEndpointGetInstance(otTcpEndpoint *aEndpoint)
Obtains the otInstance that was associated with `aEndpoint` upon initialization.

void * otTcpEndpointGetContext(otTcpEndpoint *aEndpoint)
Obtains the context pointer that was associated with `aEndpoint` upon initialization.

const otSockAddr * otTcpGetLocalAddress(const otTcpEndpoint *aEndpoint)
Obtains a pointer to a TCP endpoint's local host and port.

const otSockAddr * otTcpGetPeerAddress(const otTcpEndpoint *aEndpoint)
Obtains a pointer to a TCP endpoint's peer's host and port.

otError otTcpBind(otTcpEndpoint *aEndpoint, const otSockAddr *aSockName)
Binds the TCP endpoint to an IP address and port.

otError otTcpConnect(otTcpEndpoint *aEndpoint, const otSockAddr *aSockName, uint32_t aFlags)
Records the remote host and port for this connection.

otError otTcpSendByReference(otTcpEndpoint *aEndpoint, otLinkedBuffer *aBuffer, uint32_t aFlags)
Adds data referenced by the linked buffer pointed to by `aBuffer` to the send buffer.

otError otTcpSendByExtension(otTcpEndpoint *aEndpoint, size_t aNumBytes, uint32_t aFlags)
Adds data to the send buffer by extending the length of the final otLinkedBuffer in the send buffer by the specified amount.

otError otTcpReceiveByReference(otTcpEndpoint *aEndpoint, const otLinkedBuffer **aBuffer)
Provides the application with a linked buffer chain referencing data currently in the TCP receive buffer.

otError otTcpReceiveContiguify(otTcpEndpoint *aEndpoint)
Reorganizes the receive buffer to be entirely contiguous in memory.

otError otTcpCommitReceive(otTcpEndpoint *aEndpoint, size_t aNumBytes, uint32_t aFlags)
Informs the TCP stack that the application has finished processing `aNumBytes` bytes of data at the start of the receive buffer and that the TCP stack need not continue maintaining those bytes in the receive buffer.

otError otTcpSendEndOfStream(otTcpEndpoint *aEndpoint)
Informs the connection peer that this TCP endpoint will not send more data.

otError otTcpAbort(otTcpEndpoint *aEndpoint)
Forcibly ends the TCP connection associated with this TCP endpoint.

otError otTcpEndpointDeinitialize(otTcpEndpoint *aEndpoint)
Deinitializes this TCP endpoint.

otError otTcpListenerInitialize(otInstance *aInstance, otTcpListener *aListener, const otTcpListenerInitializeArgs *aArgs)
Initializes a TCP listener.

otInstance * otTcpListenerGetInstance(otTcpListener *aListener)
Obtains the otInstance that was associated with `aListener` upon initialization.

void * otTcpListenerGetContext(otTcpListener *aListener)
Obtains the context pointer that was associated with `aListener` upon initialization.

otError otTcpListen(otTcpListener *aListener, const otSockAddr *aSockName)
Causes incoming TCP connections that match the specified IP address and port to trigger this TCP listener's callbacks.

otError otTcpStopListening(otTcpListener *aListener)
Causes this TCP listener to stop listening for incoming connections.

otError otTcpListenerDeinitialize(otTcpListener *aListener)
Deinitializes this TCP listener.

Macros

#define OT_TCP_ENDPOINT_TCB_SIZE_BASE 392
OT_TCP_ENDPOINT_TCB_SIZE_BASE and OT_TCP_ENDPOINT_TCB_NUM_POINTERS are chosen such that the mTcb field of otTcpEndpoint has the same size as struct tcpcb in TCPlp.

#define OT_TCP_ENDPOINT_TCB_NUM_PTR 36

#define OT_TCP_RECEIVE_BUFFER_SIZE_FEW_HOPS 2598
Recommended buffer size for TCP connections that traverse about 3 wireless hops or fewer.

#define OT_TCP_RECEIVE_BUFFER_SIZE_MANY_HOPS 4157
Recommended buffer size for TCP connections that traverse many wireless hops.

#define OT_TCP_LISTENER_TCB_SIZE_BASE 16
OT_TCP_LISTENER_TCB_SIZE_BASE and OT_TCP_LISTENER_TCB_NUM_POINTERS are chosen such that the mTcbListener field of otTcpListener has the same size as struct tcpcb_listen in TCPlp.

#define OT_TCP_LISTENER_TCB_NUM_PTR 3

Enumeration Documentation

otTcpDisconnectedReason

> otTcpDisconnectedReason

Enumerator
OT_TCP_DISCONNECTED_REASON_NORMAL
OT_TCP_DISCONNECTED_REASON_REFUSED
OT_TCP_DISCONNECTED_REASON_RESET
OT_TCP_DISCONNECTED_REASON_TIME_WAIT
OT_TCP_DISCONNECTED_REASON_TIMED_OUT

Definition at line `191` of file `include/openthread/tcp.h`

@22

> @22

Defines flags passed to otTcpConnect().

Enumerator
OT_TCP_CONNECT_NO_FAST_OPEN

Definition at line `396` of file `include/openthread/tcp.h`

@23

> @23

Defines flags passed to `otTcpSendByReference`.

Enumerator
OT_TCP_SEND_MORE_TO_COME

Definition at line `429` of file `include/openthread/tcp.h`

otTcpIncomingConnectionAction

> otTcpIncomingConnectionAction

Defines incoming connection actions.

This is used in otTcpAcceptReady() callback.

Enumerator	
OT_TCP_INCOMING_CONNECTION_ACTION_ACCEPT	Accept the incoming connection.
OT_TCP_INCOMING_CONNECTION_ACTION_DEFER	Defer (silently ignore) the incoming connection.
OT_TCP_INCOMING_CONNECTION_ACTION_REFUSE	Refuse the incoming connection.

Definition at line 595 of file include/openthread/tcp.h

Typedef Documentation

otLinkedBuffer

```
typedef struct otLinkedBuffer otLinkedBuffer
```

A linked buffer structure for use with TCP.

A single otLinkedBuffer structure references an array of bytes in memory, via mData and mLength. The mNext field is used to form a chain of otLinkedBuffer structures.

Definition at line 68 of file include/openthread/tcp.h

otTcpEndpoint

```
typedef struct otTcpEndpoint otTcpEndpoint
```

Definition at line 71 of file include/openthread/tcp.h

otTcpEstablished

```
typedef void(* otTcpEstablished) (otTcpEndpoint *aEndpoint) )(otTcpEndpoint *aEndpoint)
```

This callback informs the application that the TCP 3-way handshake is complete and that the connection is now established.

Parameters

[in]	aEndpoint	The TCP endpoint whose connection is now established.

Definition at line 80 of file include/openthread/tcp.h

otTcpSendDone

```
typedef void(* otTcpSendDone) (otTcpEndpoint *aEndpoint, otLinkedBuffer *aData) )(otTcpEndpoint *aEndpoint,
otLinkedBuffer *aData)
```

This callback informs the application that data in the provided aData have been acknowledged by the connection peer and that aData and the data it contains can be reclaimed by the application.

Parameters

[in]	aEndpoint	The TCP endpoint for the connection.

[in]	aData	A pointer to the otLinkedBuffer that can be reclaimed.

The `aData` are guaranteed to be identical to those passed in to TCP via otTcpSendByReference(), including any extensions effected via otTcpSendByExtension().

Definition at line `95` of file `include/openthread/tcp.h`

otTcpForwardProgress

```
typedef void(* otTcpForwardProgress) (otTcpEndpoint *aEndpoint, size_t aInSendBuffer, size_t aBacklog) )(otTcpEndpoint *aEndpoint, size_t aInSendBuffer, size_t aBacklog)
```

This callback informs the application if forward progress has been made in transferring data from the send buffer to the recipient.

Parameters

[in]	aEndpoint	The TCP endpoint for the connection.
[in]	aInSendBuffer	The number of bytes in the send buffer (sum of "in-flight" and "backlog" regions).
[in]	aBacklog	The number of bytes that are queued for sending but have not yet been sent (the "backlog" region).

This callback is not necessary for correct TCP operation. Most applications can just rely on the otTcpSendDone() callback to reclaim linked buffers once the TCP stack is done using them. The purpose of this callback is to support advanced applications that benefit from finer-grained information about how the the connection is making forward progress in transferring data to the connection peer.

This callback's operation is closely tied to TCP's send buffer. The send buffer can be understood as having two regions. First, there is the "in-flight" region at the head (front) of the send buffer. It corresponds to data which has been sent to the recipient, but is not yet acknowledged. Second, there is the "backlog" region, which consists of all data in the send buffer that is not in the "in-flight" region. The "backlog" region corresponds to data that is queued for sending, but has not yet been sent.

The callback is invoked in response to two types of events. First, the "in-flight" region of the send buffer may shrink (e.g., when the recipient acknowledges data that we sent earlier). Second, the "backlog" region of the send buffer may shrink (e.g., new data was sent out). These two conditions often occur at the same time, in response to an ACK segment from the connection peer, which is why they are combined in a single callback.

The TCP stack only uses the `aInSendBuffer` bytes at the tail of the send buffer; when `aInSendBuffer` decreases by an amount x, it means that x additional bytes that were formerly at the head of the send buffer are no longer part of the send buffer and can now be reclaimed (i.e., overwritten) by the application. Note that the otLinkedBuffer structure itself can only be reclaimed once all bytes that it references are no longer part of the send buffer.

This callback subsumes otTcpSendDone(), in the following sense: applications can determine when linked buffers can be reclaimed by comparing `aInSendBuffer` with how many bytes are in each linked buffer. However, we expect otTcpSendDone(), which directly conveys which otLinkedBuffers can be reclaimed, to be much simpler to use. If both callbacks are registered and are triggered by the same event (e.g., the same ACK segment received), then the otTcpSendDone() callback will be triggered first, followed by this callback.

Additionally, this callback provides `aBacklog`, which indicates how many bytes of data in the send buffer are not yet in flight. For applications that only want to add data to the send buffer when there is an assurance that it will be sent out soon, it may be desirable to only send out data when `aBacklog` is suitably small (0 or close to 0). For example, an application may use `aBacklog` so that it can react to queue buildup by dropping or aggregating data to avoid creating a backlog of data.

After a call to otTcpSendByReference() or otTcpSendByExtension() with a positive number of bytes, the otTcpForwardProgress() callback is guaranteed to be called, to indicate when the bytes that were added to the send buffer are sent out. The call to otTcpForwardProgress() may be made immediately after the bytes are added to the send buffer (if some of those bytes are immediately sent out, reducing the backlog), or sometime in the future (once the connection sends out some or all of the data, reducing the backlog). By "immediately," we mean that the callback is immediately

scheduled for execution in a tasklet; to avoid reentrancy-related complexity, the otTcpForwardProgress() callback is never directly called from the otTcpSendByReference() or otTcpSendByExtension() functions.

Definition at line 165 of file include/openthread/tcp.h

otTcpReceiveAvailable

```
typedef void(* otTcpReceiveAvailable) (otTcpEndpoint *aEndpoint, size_t aBytesAvailable, bool aEndOfStream, size_t aBytesRemaining) )(otTcpEndpoint *aEndpoint, size_t aBytesAvailable, bool aEndOfStream, size_t aBytesRemaining)
```

This callback indicates the number of bytes available for consumption from the receive buffer.

Parameters

[in]	aEndpoint	The TCP endpoint for the connection.
[in]	aBytesAvailable	The number of bytes in the connection's receive buffer.
[in]	aEndOfStream	Indicates if additional data, beyond what is already in the connection's receive buffer, can be received.
[in]	aBytesRemaining	The number of additional bytes that can be received before the receive buffer becomes full.

It is called whenever bytes are added to the receive buffer and when the end of stream is reached. If the end of the stream has been reached (i.e., if no more data will become available to read because the connection peer has closed their end of the connection for writing), then `aEndOfStream` is true. Finally, `aBytesRemaining` indicates how much capacity is left in the receive buffer to hold additional data that arrives.

Definition at line 186 of file include/openthread/tcp.h

otTcpDisconnectedReason

```
typedef enum otTcpDisconnectedReason otTcpDisconnectedReason
```

Definition at line 198 of file include/openthread/tcp.h

otTcpDisconnected

```
typedef void(* otTcpDisconnected) (otTcpEndpoint *aEndpoint, otTcpDisconnectedReason aReason) )(otTcpEndpoint *aEndpoint, otTcpDisconnectedReason aReason)
```

This callback indicates that the connection was broken and should no longer be used, or that a connection has entered the TIME-WAIT state.

Parameters

[in]	aEndpoint	The TCP endpoint whose connection has been lost.
[in]	aReason	The reason why the connection was lost.

It can occur if a connection establishment attempt (initiated by calling otTcpConnect()) fails, or any point thereafter (e.g., if the connection times out or an RST segment is received from the connection peer). Once this callback fires, all resources that the application provided for this connection (i.e., any `otLinkedBuffers` and memory they reference, but not the TCP endpoint itself or space for the receive buffers) can be reclaimed. In the case of a connection entering the TIME-WAIT state, this callback is called twice, once upon entry into the TIME-WAIT state (with OT_TCP_DISCONNECTED_REASON_TIME_WAIT, and again when the TIME-WAIT state expires (with OT_TCP_DISCONNECTED_REASON_NORMAL).

Definition at line 219 of file include/openthread/tcp.h

otTcpEndpointInitializeArgs

```
typedef struct otTcpEndpointInitializeArgs otTcpEndpointInitializeArgs
```

Contains arguments to the otTcpEndpointInitialize() function.

Definition at line 284 of file include/openthread/tcp.h

otTcpListener

```
typedef struct otTcpListener otTcpListener
```

Definition at line 587 of file include/openthread/tcp.h

otTcpIncomingConnectionAction

```
typedef enum otTcpIncomingConnectionAction otTcpIncomingConnectionAction
```

Defines incoming connection actions.

This is used in otTcpAcceptReady() callback.

Definition at line 600 of file include/openthread/tcp.h

otTcpAcceptReady

```
typedef otTcpIncomingConnectionAction(* otTcpAcceptReady) (otTcpListener *aListener, const otSockAddr *aPeer,
otTcpEndpoint **aAcceptInto) )(otTcpListener *aListener, const otSockAddr *aPeer, otTcpEndpoint **aAcceptInto)
```

This callback indicates that an incoming connection that matches this TCP listener has arrived.

Parameters

[in]	aListener	The TCP listener that matches the incoming connection.
[in]	aPeer	The host and port from which the incoming connection originates.
[out]	aAcceptInto	The TCP endpoint into which to accept the incoming connection.

The typical response is for the application to accept the incoming connection. It does so by populating `aAcceptInto` with a pointer to the otTcpEndpoint into which to accept the incoming connection. This otTcpEndpoint must already be initialized using otTcpEndpointInitialize(). Then, the application returns OT_TCP_INCOMING_CONNECTION_ACTION_ACCEPT.

Alternatively, the application can decline to accept the incoming connection. There are two ways for the application to do this. First, if the application returns OT_TCP_INCOMING_CONNECTION_ACTION_DEFER, then OpenThread silently ignores the connection establishment request; the connection peer will likely retransmit the request, at which point the callback will be called again. This is valuable if resources are not presently available to accept the connection, but they may be available when the connection peer retransmits its connection establishment attempt. Second, if the application returns OT_TCP_INCOMING_CONNECTION_ACTION_REFUSE, then OpenThread sends a "connection refused" message to the host that attempted to establish a connection. If the application declines the incoming connection, it is not required to populate `aAcceptInto` .

Returns

- Description of how to handle the incoming connection.

Definition at line 632 of file include/openthread/tcp.h

otTcpAcceptDone

```
typedef void(* otTcpAcceptDone) (otTcpListener *aListener, otTcpEndpoint *aEndpoint, const otSockAddr *aPeer) )
(otTcpListener *aListener, otTcpEndpoint *aEndpoint, const otSockAddr *aPeer)
```

This callback indicates that the TCP connection is now ready for two-way communication.

Parameters

[in]	aListener	The TCP listener that matches the incoming connection.
[in]	aEndpoint	The TCP endpoint into which the incoming connection was accepted.
[in]	aPeer	the host and port from which the incoming connection originated.

In the case of TCP Fast Open, this may be before the TCP connection handshake has actually completed. The application is provided with the context pointers both for the TCP listener that accepted the connection and the TCP endpoint into which it was accepted. The provided context is the one associated with the TCP listener.

Definition at line 651 of file include/openthread/tcp.h

otTcpListenerInitializeArgs

```
typedef struct otTcpListenerInitializeArgs otTcpListenerInitializeArgs
```

Contains arguments to the otTcpListenerInitialize() function.

Definition at line 698 of file include/openthread/tcp.h

Function Documentation

otTcpEndpointInitialize

```
otError otTcpEndpointInitialize (otInstance *aInstance, otTcpEndpoint *aEndpoint, const otTcpEndpointInitializeArgs
*aArgs)
```

Initializes a TCP endpoint.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEndpoint	A pointer to a TCP endpoint structure.
[in]	aArgs	A pointer to a structure of arguments.

Calling this function causes OpenThread to keep track of the TCP endpoint and store and retrieve TCP data inside the `aEndpoint`. The application should refrain from directly accessing or modifying the fields in `aEndpoint`. If the application needs to reclaim the memory backing `aEndpoint`, it should call otTcpEndpointDeinitialize().

Definition at line 328 of file include/openthread/tcp.h

otTcpEndpointGetInstance

```
otInstance * otTcpEndpointGetInstance (otTcpEndpoint *aEndpoint)
```

Obtains the otInstance that was associated with `aEndpoint` upon initialization.

Parameters

[in]	aEndpoint	The TCP endpoint whose instance to obtain.

Returns

- The otInstance pointer associated with `aEndpoint`.

Definition at line `341` of file `include/openthread/tcp.h`

otTcpEndpointGetContext

```
void * otTcpEndpointGetContext (otTcpEndpoint *aEndpoint)
```

Obtains the context pointer that was associated with `aEndpoint` upon initialization.

Parameters

[in]	aEndpoint	The TCP endpoint whose context to obtain.

Returns

- The context pointer associated with `aEndpoint`.

Definition at line `352` of file `include/openthread/tcp.h`

otTcpGetLocalAddress

```
const otSockAddr * otTcpGetLocalAddress (const otTcpEndpoint *aEndpoint)
```

Obtains a pointer to a TCP endpoint's local host and port.

Parameters

[in]	aEndpoint	The TCP endpoint whose local host and port to obtain.

The contents of the host and port may be stale if this socket is not in a connected state and has not been bound after it was last disconnected.

Returns

- The local host and port of `aEndpoint`.

Definition at line `365` of file `include/openthread/tcp.h`

otTcpGetPeerAddress

```
const otSockAddr * otTcpGetPeerAddress (const otTcpEndpoint *aEndpoint)
```

Obtains a pointer to a TCP endpoint's peer's host and port.

Parameters

[in]	aEndpoint	The TCP endpoint whose peer's host and port to obtain.

The contents of the host and port may be stale if this socket is not in a connected state.

Returns

- The host and port of the connection peer of `aEndpoint`.

Definition at line `378` of file `include/openthread/tcp.h`

otTcpBind

```
otError otTcpBind (otTcpEndpoint *aEndpoint, const otSockAddr *aSockName)
```

Binds the TCP endpoint to an IP address and port.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint structure to bind.
[in]	aSockName	The address and port to which to bind this TCP endpoint.

Definition at line `390` of file `include/openthread/tcp.h`

otTcpConnect

```
otError otTcpConnect (otTcpEndpoint *aEndpoint, const otSockAddr *aSockName, uint32_t aFlags)
```

Records the remote host and port for this connection.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint structure to connect.
[in]	aSockName	The IP address and port of the host to which to connect.
[in]	aFlags	Flags specifying options for this operation (see enumeration above).

TCP Fast Open must be enabled or disabled using `aFlags`. If it is disabled, then the TCP connection establishment handshake is initiated immediately. If it is enabled, then this function merely records the the remote host and port, and the TCP connection establishment handshake only happens on the first call to `otTcpSendByReference()`.

If TCP Fast Open is disabled, then the caller must wait for the `otTcpEstablished` callback indicating that TCP connection establishment handshake is done before it can start sending data e.g., by calling `otTcpSendByReference()`.

Definition at line `423` of file `include/openthread/tcp.h`

otTcpSendByReference

```
otError otTcpSendByReference (otTcpEndpoint *aEndpoint, otLinkedBuffer *aBuffer, uint32_t aFlags)
```

Adds data referenced by the linked buffer pointed to by `aBuffer` to the send buffer.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint structure representing the TCP endpoint on which to send data.
[in]	aBuffer	A pointer to the linked buffer chain referencing data to add to the send buffer.
[in]	aFlags	Flags specifying options for this operation (see enumeration above).

Upon a successful call to this function, the linked buffer and data it references are owned by the TCP stack; they should not be modified by the application until a "send done" callback returns ownership of those objects to the application. It is

acceptable to call this function to add another linked buffer to the send queue, even if the "send done" callback for a previous invocation of this function has not yet fired.

Note that `aBuffer` should not be chained; its mNext field should be NULL. If additional data will be added right after this call, then the OT_TCP_SEND_MORE_TO_COME flag should be used as a hint to the TCP implementation.

Definition at line `458` of file `include/openthread/tcp.h`

otTcpSendByExtension

```
otError otTcpSendByExtension (otTcpEndpoint *aEndpoint, size_t aNumBytes, uint32_t aFlags)
```

Adds data to the send buffer by extending the length of the final otLinkedBuffer in the send buffer by the specified amount.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint structure representing the TCP endpoint on which to send data.
[in]	aNumBytes	The number of bytes by which to extend the length of the final linked buffer.
[in]	aFlags	Flags specifying options for this operation (see enumeration above).

If the send buffer is empty, then the operation fails.

Definition at line `474` of file `include/openthread/tcp.h`

otTcpReceiveByReference

```
otError otTcpReceiveByReference (otTcpEndpoint *aEndpoint, const otLinkedBuffer **aBuffer)
```

Provides the application with a linked buffer chain referencing data currently in the TCP receive buffer.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint structure representing the TCP endpoint on which to receive data.
[out]	aBuffer	A pointer to the linked buffer chain referencing data currently in the receive buffer.

The linked buffer chain is valid until the "receive ready" callback is next invoked, or until the next call to otTcpReceiveContiguify() or otTcpCommitReceive().

Definition at line `492` of file `include/openthread/tcp.h`

otTcpReceiveContiguify

```
otError otTcpReceiveContiguify (otTcpEndpoint *aEndpoint)
```

Reorganizes the receive buffer to be entirely contiguous in memory.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint whose receive buffer to reorganize.

This is optional; an application can simply traverse the linked buffer chain obtained by calling `otTcpReceiveByReference`. Some applications may wish to call this function to make the receive buffer contiguous to simplify their data processing, but this comes at the expense of CPU time to reorganize the data in the receive buffer.

Definition at line `509` of file `include/openthread/tcp.h`

otTcpCommitReceive

```
otError otTcpCommitReceive (otTcpEndpoint *aEndpoint, size_t aNumBytes, uint32_t aFlags)
```

Informs the TCP stack that the application has finished processing `aNumBytes` bytes of data at the start of the receive buffer and that the TCP stack need not continue maintaining those bytes in the receive buffer.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint structure representing the TCP endpoint on which to receive data.
[in]	aNumBytes	The number of bytes consumed.
[in]	aFlags	Flags specifying options for this operation (none yet).

Definition at line `525` of file `include/openthread/tcp.h`

otTcpSendEndOfStream

```
otError otTcpSendEndOfStream (otTcpEndpoint *aEndpoint)
```

Informs the connection peer that this TCP endpoint will not send more data.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint structure representing the TCP endpoint to shut down.

This should be used when the application has no more data to send to the connection peer. For this connection, future reads on the connection peer will result in the "end of stream" condition, and future writes on this connection endpoint will fail.

The "end of stream" condition only applies after any data previously provided to the TCP stack to send out has been received by the connection peer.

Definition at line `545` of file `include/openthread/tcp.h`

otTcpAbort

```
otError otTcpAbort (otTcpEndpoint *aEndpoint)
```

Forcibly ends the TCP connection associated with this TCP endpoint.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint structure representing the TCP endpoint to abort.

This immediately makes the TCP endpoint free for use for another connection and empties the send and receive buffers, transferring ownership of any data provided by the application in otTcpSendByReference() and otTcpSendByExtension() calls back to the application. The TCP endpoint's callbacks and memory for the receive buffer remain associated with the TCP endpoint.

Definition at line `563` of file `include/openthread/tcp.h`

otTcpEndpointDeinitialize

```
otError otTcpEndpointDeinitialize (otTcpEndpoint *aEndpoint)
```

Deinitializes this TCP endpoint.

Parameters

[in]	aEndpoint	A pointer to the TCP endpoint structure to deinitialize.

This means that OpenThread no longer keeps track of this TCP endpoint and deallocates all resources it has internally allocated for this TCP endpoint. The application can reuse the memory backing the TCP endpoint as it sees fit.

If it corresponds to a live TCP connection, the connection is terminated unceremoniously (as in otTcpAbort()). All resources the application has provided for this TCP endpoint (linked buffers for the send buffer, memory for the receive buffer, the `aEndpoint` structure itself, etc.) are immediately returned to the application.

Definition at line `584` of file `include/openthread/tcp.h`

otTcpListenerInitialize

> otError otTcpListenerInitialize (otInstance *aInstance, otTcpListener *aListener, const otTcpListenerInitializeArgs *aArgs)

Initializes a TCP listener.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aListener	A pointer to a TCP listener structure.
[in]	aArgs	A pointer to a structure of arguments.

Calling this function causes OpenThread to keep track of the TCP listener and store and retrieve TCP data inside `aListener`. The application should refrain from directly accessing or modifying the fields in `aListener`. If the application needs to reclaim the memory backing `aListener`, it should call otTcpListenerDeinitialize().

Definition at line `717` of file `include/openthread/tcp.h`

otTcpListenerGetInstance

> otInstance * otTcpListenerGetInstance (otTcpListener *aListener)

Obtains the otInstance that was associated with `aListener` upon initialization.

Parameters

[in]	aListener	The TCP listener whose instance to obtain.

Returns

- The otInstance pointer associated with `aListener`.

Definition at line `730` of file `include/openthread/tcp.h`

otTcpListenerGetContext

> void * otTcpListenerGetContext (otTcpListener *aListener)

Obtains the context pointer that was associated with `aListener` upon initialization.

Parameters

[in]	aListener	The TCP listener whose context to obtain.

Returns

- The context pointer associated with `aListener`.

Definition at line `741` of file `include/openthread/tcp.h`

otTcpListen

```
otError otTcpListen (otTcpListener *aListener, const otSockAddr *aSockName)
```

Causes incoming TCP connections that match the specified IP address and port to trigger this TCP listener's callbacks.

Parameters

[in]	aListener	A pointer to the TCP listener structure that should begin listening.
[in]	aSockName	The address and port on which to listen for incoming connections.

Definition at line `754` of file `include/openthread/tcp.h`

otTcpStopListening

```
otError otTcpStopListening (otTcpListener *aListener)
```

Causes this TCP listener to stop listening for incoming connections.

Parameters

[in]	aListener	A pointer to the TCP listener structure that should stop listening.

Definition at line `765` of file `include/openthread/tcp.h`

otTcpListenerDeinitialize

```
otError otTcpListenerDeinitialize (otTcpListener *aListener)
```

Deinitializes this TCP listener.

Parameters

[in]	aListener	A pointer to the TCP listener structure to deinitialize.

This means that OpenThread no longer keeps track of this TCP listener and deallocates all resources it has internally allocated for this TCP listener. The application can reuse the memory backing the TCP listener as it sees fit.

If the TCP listener is currently listening, it stops listening.

Definition at line `783` of file `include/openthread/tcp.h`

Macro Definition Documentation

OT_TCP_ENDPOINT_TCB_SIZE_BASE

```
#define OT_TCP_ENDPOINT_TCB_SIZE_BASE
```

Value:

```
392
```

OT_TCP_ENDPOINT_TCB_SIZE_BASE and OT_TCP_ENDPOINT_TCB_NUM_POINTERS are chosen such that the mTcb field of otTcpEndpoint has the same size as struct tcpcb in TCPlp.

This is necessary because the mTcb field, although opaque in its declaration, is treated as struct tcpcb in the TCP implementation.

Definition at line 228 of file include/openthread/tcp.h

OT_TCP_ENDPOINT_TCB_NUM_PTR

```
#define OT_TCP_ENDPOINT_TCB_NUM_PTR
```

Value:

```
36
```

Definition at line 229 of file include/openthread/tcp.h

OT_TCP_RECEIVE_BUFFER_SIZE_FEW_HOPS

```
#define OT_TCP_RECEIVE_BUFFER_SIZE_FEW_HOPS
```

Value:

```
2598
```

Recommended buffer size for TCP connections that traverse about 3 wireless hops or fewer.

On platforms where memory is particularly constrained and in situations where high bandwidth is not necessary, it may be desirable to manually select a smaller buffer size.

Definition at line 297 of file include/openthread/tcp.h

OT_TCP_RECEIVE_BUFFER_SIZE_MANY_HOPS

```
#define OT_TCP_RECEIVE_BUFFER_SIZE_MANY_HOPS
```

Value:

```
4157
```

Recommended buffer size for TCP connections that traverse many wireless hops.

If the TCP connection traverses a very large number of hops (more than 6 or so), then it may be advisable to select a large buffer size manually.

Definition at line 309 of file include/openthread/tcp.h

OT_TCP_LISTENER_TCB_SIZE_BASE

```
#define OT_TCP_LISTENER_TCB_SIZE_BASE
```

Value:

```
16
```

OT_TCP_LISTENER_TCB_SIZE_BASE and OT_TCP_LISTENER_TCB_NUM_POINTERS are chosen such that the mTcbListener field of otTcpListener has the same size as struct tcpcb_listen in TCPlp.

This is necessary because the mTcbListen field, though opaque in its declaration, is treated as struct tcpcb in the TCP implementation.

Definition at line `660` of file `include/openthread/tcp.h`

OT_TCP_LISTENER_TCB_NUM_PTR

```
#define OT_TCP_LISTENER_TCB_NUM_PTR
```

Value:

```
3
```

Definition at line `661` of file `include/openthread/tcp.h`

otLinkedBuffer

A linked buffer structure for use with TCP.

A single otLinkedBuffer structure references an array of bytes in memory, via mData and mLength. The mNext field is used to form a chain of otLinkedBuffer structures.

Public Attributes

struct otLinkedBuffer *	mNext Pointer to the next linked buffer in the chain, or NULL if it is the end.
const uint8_t *	mData Pointer to data referenced by this linked buffer.
size_t	mLength Length of this linked buffer (number of bytes).

Public Attribute Documentation

mNext

```
struct otLinkedBuffer* otLinkedBuffer::mNext
```

Pointer to the next linked buffer in the chain, or NULL if it is the end.

Definition at line `65` of file `include/openthread/tcp.h`

mData

```
const uint8_t* otLinkedBuffer::mData
```

Pointer to data referenced by this linked buffer.

Definition at line `66` of file `include/openthread/tcp.h`

mLength

```
size_t otLinkedBuffer::mLength
```

Length of this linked buffer (number of bytes).

Definition at line `67` of file `include/openthread/tcp.h`

otTcpEndpoint

Represents a TCP endpoint.

An TCP endpoint acts an endpoint of TCP connection. It can be used to initiate TCP connections, and, once a TCP connection is established, send data to and receive data from the connection peer.

The application should not inspect the fields of this structure directly; it should only interact with it via the TCP API functions whose signatures are provided in this file.

Public Attributes

uint8_t	mSize
uint64_t	mAlign
union otTcpEndpoint::@24	mTcb
struct otTcpEndpoint *	mNext A pointer to the next TCP endpoint (internal use only)
void *	mContext A pointer to application-specific context.
otTcpEstablished	mEstablishedCallback "Established" callback function
otTcpSendDone	mSendDoneCallback "Send done" callback function
otTcpForwardProgress	mForwardProgressCallback "Forward progress" callback function
otTcpReceiveAvailable	mReceiveAvailableCallback "Receive available" callback function
otTcpDisconnected	mDisconnectedCallback "Disconnected" callback function
uint32_t	mTimers
otLinkedBuffer	mReceiveLinks
otSockAddr	mSockAddr
uint8_t	mPendingCallbacks

Public Attribute Documentation

mSize

> uint8_t otTcpEndpoint::mSize[OT_TCP_ENDPOINT_TCB_SIZE_BASE+OT_TCP_ENDPOINT_TCB_NUM_PTR *sizeof(void *)]

Definition at line 247 of file include/openthread/tcp.h

mAlign

```
uint64_t otTcpEndpoint::mAlign
```

Definition at line 248 of file include/openthread/tcp.h

mTcb

```
union otTcpEndpoint::@24 otTcpEndpoint::mTcb
```

Definition at line 249 of file include/openthread/tcp.h

mNext

```
struct otTcpEndpoint* otTcpEndpoint::mNext
```

A pointer to the next TCP endpoint (internal use only)

Definition at line 251 of file include/openthread/tcp.h

mContext

```
void* otTcpEndpoint::mContext
```

A pointer to application-specific context.

Definition at line 252 of file include/openthread/tcp.h

mEstablishedCallback

```
otTcpEstablished otTcpEndpoint::mEstablishedCallback
```

"Established" callback function

Definition at line 254 of file include/openthread/tcp.h

mSendDoneCallback

```
otTcpSendDone otTcpEndpoint::mSendDoneCallback
```

"Send done" callback function

Definition at line 255 of file include/openthread/tcp.h

mForwardProgressCallback

> otTcpForwardProgress otTcpEndpoint::mForwardProgressCallback

"Forward progress" callback function

Definition at line `256` of file `include/openthread/tcp.h`

mReceiveAvailableCallback

> otTcpReceiveAvailable otTcpEndpoint::mReceiveAvailableCallback

"Receive available" callback function

Definition at line `257` of file `include/openthread/tcp.h`

mDisconnectedCallback

> otTcpDisconnected otTcpEndpoint::mDisconnectedCallback

"Disconnected" callback function

Definition at line `258` of file `include/openthread/tcp.h`

mTimers

> uint32_t otTcpEndpoint::mTimers[4]

Definition at line `260` of file `include/openthread/tcp.h`

mReceiveLinks

> otLinkedBuffer otTcpEndpoint::mReceiveLinks[2]

Definition at line `262` of file `include/openthread/tcp.h`

mSockAddr

> otSockAddr otTcpEndpoint::mSockAddr

Definition at line `263` of file `include/openthread/tcp.h`

mPendingCallbacks

> uint8_t otTcpEndpoint::mPendingCallbacks

Definition at line `265` of file `include/openthread/tcp.h`

otTcpEndpointInitializeArgs

Contains arguments to the otTcpEndpointInitialize() function.

Public Attributes

void *	mContext Pointer to application-specific context.
otTcpEstablished	mEstablishedCallback "Established" callback function
otTcpSendDone	mSendDoneCallback "Send done" callback function
otTcpForwardProgress	mForwardProgressCallback "Forward progress" callback function
otTcpReceiveAvailable	mReceiveAvailableCallback "Receive available" callback function
otTcpDisconnected	mDisconnectedCallback "Disconnected" callback function
void *	mReceiveBuffer Pointer to memory provided to the system for the TCP receive buffer.
size_t	mReceiveBufferSize Size of memory provided to the system for the TCP receive buffer.

Public Attribute Documentation

mContext

```
void* otTcpEndpointInitializeArgs::mContext
```

Pointer to application-specific context.

Definition at line 274 of file include/openthread/tcp.h

mEstablishedCallback

```
otTcpEstablished otTcpEndpointInitializeArgs::mEstablishedCallback
```

"Established" callback function

Definition at line 276 of file include/openthread/tcp.h

mSendDoneCallback

```
otTcpSendDone otTcpEndpointInitializeArgs::mSendDoneCallback
```

"Send done" callback function

Definition at line 277 of file include/openthread/tcp.h

mForwardProgressCallback

> otTcpForwardProgress otTcpEndpointInitializeArgs::mForwardProgressCallback

"Forward progress" callback function

Definition at line 278 of file include/openthread/tcp.h

mReceiveAvailableCallback

> otTcpReceiveAvailable otTcpEndpointInitializeArgs::mReceiveAvailableCallback

"Receive available" callback function

Definition at line 279 of file include/openthread/tcp.h

mDisconnectedCallback

> otTcpDisconnected otTcpEndpointInitializeArgs::mDisconnectedCallback

"Disconnected" callback function

Definition at line 280 of file include/openthread/tcp.h

mReceiveBuffer

> void* otTcpEndpointInitializeArgs::mReceiveBuffer

Pointer to memory provided to the system for the TCP receive buffer.

Definition at line 282 of file include/openthread/tcp.h

mReceiveBufferSize

> size_t otTcpEndpointInitializeArgs::mReceiveBufferSize

Size of memory provided to the system for the TCP receive buffer.

Definition at line 283 of file include/openthread/tcp.h

otTcpListener

Represents a TCP listener.

A TCP listener is used to listen for and accept incoming TCP connections.

The application should not inspect the fields of this structure directly; it should only interact with it via the TCP API functions whose signatures are provided in this file.

Public Attributes

uint8_t	mSize
void *	mAlign
union otTcpListener::@25	mTcbListen
struct otTcpListener *	mNext A pointer to the next TCP listener (internal use only)
void *	mContext A pointer to application-specific context.
otTcpAcceptReady	mAcceptReadyCallback "Accept ready" callback function
otTcpAcceptDone	mAcceptDoneCallback "Accept done" callback function

Public Attribute Documentation

mSize

```
uint8_t otTcpListener::mSize[OT_TCP_LISTENER_TCB_SIZE_BASE+OT_TCP_LISTENER_TCB_NUM_PTR *sizeof(void *)]
```

Definition at line `677` of file `include/openthread/tcp.h`

mAlign

```
void* otTcpListener::mAlign
```

Definition at line `678` of file `include/openthread/tcp.h`

mTcbListen

```
union otTcpListener::@25 otTcpListener::mTcbListen
```

Definition at line `679` of file `include/openthread/tcp.h`

mNext

```
struct otTcpListener* otTcpListener::mNext
```

A pointer to the next TCP listener (internal use only)

Definition at line `681` of file `include/openthread/tcp.h`

mContext

```
void* otTcpListener::mContext
```

A pointer to application-specific context.

Definition at line `682` of file `include/openthread/tcp.h`

mAcceptReadyCallback

```
otTcpAcceptReady otTcpListener::mAcceptReadyCallback
```

"Accept ready" callback function

Definition at line `684` of file `include/openthread/tcp.h`

mAcceptDoneCallback

```
otTcpAcceptDone otTcpListener::mAcceptDoneCallback
```

"Accept done" callback function

Definition at line `685` of file `include/openthread/tcp.h`

otTcpListenerInitializeArgs

Contains arguments to the otTcpListenerInitialize() function.

Public Attributes

void *	mContext Pointer to application-specific context.
otTcpAcceptReady	mAcceptReadyCallback "Accept ready" callback function
otTcpAcceptDone	mAcceptDoneCallback "Accept done" callback function

Public Attribute Documentation

mContext

```
void* otTcpListenerInitializeArgs::mContext
```

Pointer to application-specific context.

Definition at line `694` of file `include/openthread/tcp.h`

mAcceptReadyCallback

```
otTcpAcceptReady otTcpListenerInitializeArgs::mAcceptReadyCallback
```

"Accept ready" callback function

Definition at line `696` of file `include/openthread/tcp.h`

mAcceptDoneCallback

```
otTcpAcceptDone otTcpListenerInitializeArgs::mAcceptDoneCallback
```

"Accept done" callback function

Definition at line `697` of file `include/openthread/tcp.h`

TCP Abstractions

TCP Abstractions

This module includes easy-to-use abstractions on top of the base TCP API.

Modules

otTcpCircularSendBuffer

otTcpEndpointAndCircularSendBuffer

Enumerations

enum @26 {

OT_TCP_CIRCULAR_SEND_BUFFER_WRITE_MORE_TO_COME = 1 << 0

}

Defines flags passed to `otTcpCircularSendBufferWrite`.

Typedefs

typedef struct otTcpCircularSendBuffer — otTcpCircularSendBuffer

Represents a circular send buffer for use with a TCP endpoint.

typedef struct otTcpEndpointAndCircularSendBuffer — otTcpEndpointAndCircularSendBuffer

Context structure to use with mbedtls_ssl_set_bio.

Functions

void otTcpCircularSendBufferInitialize(otTcpCircularSendBuffer *aSendBuffer, void *aDataBuffer, size_t aCapacity)

Initializes a TCP circular send buffer.

otError otTcpCircularSendBufferWrite(otTcpEndpoint *aEndpoint, otTcpCircularSendBuffer *aSendBuffer, const void *aData, size_t aLength, size_t *aWritten, uint32_t aFlags)

Sends out data on a TCP endpoint, using the provided TCP circular send buffer to manage buffering.

void otTcpCircularSendBufferHandleForwardProgress(otTcpCircularSendBuffer *aSendBuffer, size_t aInSendBuffer)

Performs circular-send-buffer-specific handling in the otTcpForwardProgress callback.

size_t otTcpCircularSendBufferGetFreeSpace(const otTcpCircularSendBuffer *aSendBuffer)

Returns the amount of free space in the TCP circular send buffer.

void otTcpCircularSendBufferForceDiscardAll(otTcpCircularSendBuffer *aSendBuffer)

Forcibly discards all data in the circular send buffer.

otError otTcpCircularSendBufferDeinitialize(otTcpCircularSendBuffer *aSendBuffer)

Deinitializes a TCP circular send buffer, detaching it if attached.

int otTcpMbedTlsSslSendCallback(void *aCtx, const unsigned char *aBuf, size_t aLen)
Non-blocking send callback to pass to mbedtls_ssl_set_bio.

int otTcpMbedTlsSslRecvCallback(void *aCtx, unsigned char *aBuf, size_t aLen)
Non-blocking receive callback to pass to mbedtls_ssl_set_bio.

Enumeration Documentation

@26

```
@26
```

Defines flags passed to `otTcpCircularSendBufferWrite`.

Enumerator
OT_TCP_CIRCULAR_SEND_BUFFER_WRITE_MORE_TO_COME

Definition at line `114` of file `include/openthread/tcp_ext.h`

Typedef Documentation

otTcpCircularSendBuffer

```
typedef struct otTcpCircularSendBuffer otTcpCircularSendBuffer
```

Represents a circular send buffer for use with a TCP endpoint.

Using a circular send buffer is optional. Applications can use a TCP endpoint to send data by managing otLinkedBuffers directly. However, some applications may find it more convenient to have a circular send buffer; such applications can call otTcpCircularSendBufferWrite() to "attach" a circular send buffer to a TCP endpoint and send out data on that TCP endpoint, relying on the circular send buffer to manage the underlying otLinkedBuffers.

otTcpCircularSendBuffer is implemented on top of the otLinkedBuffer-based API provided by an otTcpEndpoint. Once attached to an otTcpEndpoint, an otTcpCircularSendBuffer performs all the work of managing otLinkedBuffers for the connection. This means that, once an otTcpCircularSendBuffer is attached to an otTcpEndpoint, the application should not call otTcpSendByReference() or otTcpSendByExtension() on that otTcpEndpoint. Instead, the application should use otTcpCircularSendBufferWrite() to add data to the send buffer.

The otTcpForwardProgress() callback is the intended way for users to learn when space becomes available in the circular send buffer. On an otTcpEndpoint to which an otTcpCircularSendBuffer is attached, the application MUST install an otTcpForwardProgress() callback and call otTcpCircularSendBufferHandleForwardProgress() on the attached otTcpCircularSendBuffer at the start of the callback function. It is recommended that the user NOT install an otTcpSendDone() callback, as all management of otLinkedBuffers is handled by the circular send buffer.

The application should not inspect the fields of this structure directly; it should only interact with it via the TCP Circular Send Buffer API functions whose signature are provided in this file.

Definition at line `98` of file `include/openthread/tcp_ext.h`

otTcpEndpointAndCircularSendBuffer

```
typedef struct otTcpEndpointAndCircularSendBuffer otTcpEndpointAndCircularSendBuffer
```

Context structure to use with mbedtls_ssl_set_bio.

Definition at line `225` of file `include/openthread/tcp_ext.h`

Function Documentation

otTcpCircularSendBufferInitialize

> void otTcpCircularSendBufferInitialize (otTcpCircularSendBuffer *aSendBuffer, void *aDataBuffer, size_t aCapacity)

Initializes a TCP circular send buffer.

Parameters

[in]	aSendBuffer	A pointer to the TCP circular send buffer to initialize.
[in]	aDataBuffer	A pointer to memory to use to store data in the TCP circular send buffer.
[in]	aCapacity	The capacity, in bytes, of the TCP circular send buffer, which must equal the size of the memory pointed to by `aDataBuffer` .

Definition at line `108` of file `include/openthread/tcp_ext.h`

otTcpCircularSendBufferWrite

> otError otTcpCircularSendBufferWrite (otTcpEndpoint *aEndpoint, otTcpCircularSendBuffer *aSendBuffer, const void *aData, size_t aLength, size_t *aWritten, uint32_t aFlags)

Sends out data on a TCP endpoint, using the provided TCP circular send buffer to manage buffering.

Parameters

[in]	aEndpoint	The TCP endpoint on which to send out data.
[in]	aSendBuffer	The TCP circular send buffer into which to copy data.
[in]	aData	A pointer to data to copy into the TCP circular send buffer.
[in]	aLength	The length of the data pointed to by `aData` to copy into the TCP circular send buffer.
[out]	aWritten	Populated with the amount of data copied into the send buffer, which might be less than `aLength` if the send buffer reaches capacity.
[in]	aFlags	Flags specifying options for this operation (see enumeration above).

Once this function is called, `aSendBuffer` and `aEndpoint` are considered "attached" to each other. While they are attached, ALL send operations for `aEndpoint` must be made using `aSendBuffer` and ALL operations on `aSendBuffer` must be associated with `aEndpoint` .

The only way to "detach" a TCP circular send buffer and a TCP endpoint is to wait for the send buffer to become completely empty. This can happen in two ways: (1) all data in the send buffer is sent and acknowledged in the normal course of TCP protocol operation, or (2) the connection is terminated.

The recommended usage pattern is to use a single TCP circular send buffer with a TCP endpoint, and to send data on that TCP endpoint only via its associated TCP circular buffer. This recommended usage pattern sidesteps the issues described above by always using a TCP endpoint and TCP circular send buffer together.

If the circular send buffer reaches capacity, only a prefix of the provided data is copied into the circular send buffer.

Definition at line `153` of file `include/openthread/tcp_ext.h`

otTcpCircularSendBufferHandleForwardProgress

> void otTcpCircularSendBufferHandleForwardProgress (otTcpCircularSendBuffer *aSendBuffer, size_t aInSendBuffer)

Performs circular-send-buffer-specific handling in the otTcpForwardProgress callback.

Parameters

[in]	aSendBuffer	A pointer to the TCP circular send buffer for the endpoint for which otTcpForwardProgress() was invoked.
[in]	aInSendBuffer	Value of `aInSendBuffer` passed to the otTcpForwardProgress() callback.

The application is expected to install an otTcpForwardProgress() callback on the otTcpEndpoint, and call this function at the start of the callback function for circular-send-buffer-specific processing.

In the callback function, the application can determine the amount of free space in the circular send buffer by calling otTcpCircularSendBufferFreeSpace(), or by comparing `aInSendBuffer` with the send buffer's capacity, chosen by the user when calling otTcpCircularSendBufferInitialize().

Definition at line `178` of file `include/openthread/tcp_ext.h`

otTcpCircularSendBufferGetFreeSpace

```
size_t otTcpCircularSendBufferGetFreeSpace (const otTcpCircularSendBuffer *aSendBuffer)
```

Returns the amount of free space in the TCP circular send buffer.

Parameters

[in]	aSendBuffer	A pointer to the TCP circular send buffer whose amount of free space to return.

This operation will always succeed.

Returns

- The amount of free space in the send buffer.

Definition at line `189` of file `include/openthread/tcp_ext.h`

otTcpCircularSendBufferForceDiscardAll

```
void otTcpCircularSendBufferForceDiscardAll (otTcpCircularSendBuffer *aSendBuffer)
```

Forcibly discards all data in the circular send buffer.

Parameters

[in]	aSendBuffer	The TCP circular send buffer whose data to discard.

The application is expected to call this function when a TCP connection is terminated unceremoniously (e.g., if the application calls otTcpEndpointAbort() or is informed of a reset connection via the otTcpConnectionLost() callback).

Calling this function on a nonempty TCP circular send buffer attached to a TCP endpoint results in undefined behavior.

Definition at line `204` of file `include/openthread/tcp_ext.h`

otTcpCircularSendBufferDeinitialize

```
otError otTcpCircularSendBufferDeinitialize (otTcpCircularSendBuffer *aSendBuffer)
```

Deinitializes a TCP circular send buffer, detaching it if attached.

Parameters

[in]	aSendBuffer	The TCP circular send buffer to deinitialize.

If the TCP circular send buffer is not empty, then this operation will fail.

Definition at line 216 of file include/openthread/tcp_ext.h

otTcpMbedTlsSslSendCallback

> int otTcpMbedTlsSslSendCallback (void *aCtx, const unsigned char *aBuf, size_t aLen)

Non-blocking send callback to pass to mbedtls_ssl_set_bio.

Parameters

[in]	aCtx	A pointer to an otTcpEndpointAndCircularSendBuffer.
[in]	aBuf	The data to add to the send buffer.
[in]	aLen	The amount of data to add to the send buffer.

Returns

- The number of bytes sent, or an mbedtls error code.

Definition at line 236 of file include/openthread/tcp_ext.h

otTcpMbedTlsSslRecvCallback

> int otTcpMbedTlsSslRecvCallback (void *aCtx, unsigned char *aBuf, size_t aLen)

Non-blocking receive callback to pass to mbedtls_ssl_set_bio.

Parameters

[in]	aCtx	A pointer to an otTcpEndpointAndCircularSendBuffer.
[out]	aBuf	The buffer into which to receive data.
[in]	aLen	The maximum amount of data that can be received.

Returns

- The number of bytes received, or an mbedtls error code.

Definition at line 247 of file include/openthread/tcp_ext.h

otTcpCircularSendBuffer

Represents a circular send buffer for use with a TCP endpoint.

Using a circular send buffer is optional. Applications can use a TCP endpoint to send data by managing otLinkedBuffers directly. However, some applications may find it more convenient to have a circular send buffer; such applications can call otTcpCircularSendBufferWrite() to "attach" a circular send buffer to a TCP endpoint and send out data on that TCP endpoint, relying on the circular send buffer to manage the underlying otLinkedBuffers.

otTcpCircularSendBuffer is implemented on top of the otLinkedBuffer-based API provided by an otTcpEndpoint. Once attached to an otTcpEndpoint, an otTcpCircularSendBuffer performs all the work of managing otLinkedBuffers for the connection. This means that, once an otTcpCircularSendBuffer is attached to an otTcpEndpoint, the application should not call otTcpSendByReference() or otTcpSendByExtension() on that otTcpEndpoint. Instead, the application should use otTcpCircularSendBufferWrite() to add data to the send buffer.

The otTcpForwardProgress() callback is the intended way for users to learn when space becomes available in the circular send buffer. On an otTcpEndpoint to which an otTcpCircularSendBuffer is attached, the application MUST install an otTcpForwardProgress() callback and call otTcpCircularSendBufferHandleForwardProgress() on the attached otTcpCircularSendBuffer at the start of the callback function. It is recommended that the user NOT install an otTcpSendDone() callback, as all management of otLinkedBuffers is handled by the circular send buffer.

The application should not inspect the fields of this structure directly; it should only interact with it via the TCP Circular Send Buffer API functions whose signature are provided in this file.

Public Attributes

uint8_t *	mDataBuffer Pointer to data in the circular send buffer.
size_t	mCapacity Length of the circular send buffer.
size_t	mStartIndex Index of the first valid byte in the send buffer.
size_t	mCapacityUsed Number of bytes stored in the send buffer.
otLinkedBuffer	mSendLinks
uint8_t	mFirstSendLinkIndex

Public Attribute Documentation

mDataBuffer

```
uint8_t* otTcpCircularSendBuffer::mDataBuffer
```

Pointer to data in the circular send buffer.

Definition at line 91 of file include/openthread/tcp_ext.h

mCapacity

```
size_t otTcpCircularSendBuffer::mCapacity
```

Length of the circular send buffer.

Definition at line `92` of file `include/openthread/tcp_ext.h`

mStartIndex

```
size_t otTcpCircularSendBuffer::mStartIndex
```

Index of the first valid byte in the send buffer.

Definition at line `93` of file `include/openthread/tcp_ext.h`

mCapacityUsed

```
size_t otTcpCircularSendBuffer::mCapacityUsed
```

Number of bytes stored in the send buffer.

Definition at line `94` of file `include/openthread/tcp_ext.h`

mSendLinks

```
otLinkedBuffer otTcpCircularSendBuffer::mSendLinks[2]
```

Definition at line `96` of file `include/openthread/tcp_ext.h`

mFirstSendLinkIndex

```
uint8_t otTcpCircularSendBuffer::mFirstSendLinkIndex
```

Definition at line `97` of file `include/openthread/tcp_ext.h`

otTcpEndpointAndCircularSendBuffer

Context structure to use with mbedtls_ssl_set_bio.

Public Attributes

otTcpEndpoint *	mEndpoint
otTcpCircularSendBuffer *	mSendBuffer

Public Attribute Documentation

mEndpoint

```
otTcpEndpoint* otTcpEndpointAndCircularSendBuffer::mEndpoint
```

Definition at line 223 of file include/openthread/tcp_ext.h

mSendBuffer

```
otTcpCircularSendBuffer* otTcpEndpointAndCircularSendBuffer::mSendBuffer
```

Definition at line 224 of file include/openthread/tcp_ext.h

UDP

UDP

Modules

- UDP
- UDP Forward

UDP

UDP

This module includes functions that control UDP communication.

Modules

otUdpReceiver

otUdpSocket

Enumerations

enum otNetifIdentifier {

OT_NETIF_UNSPECIFIED = 0
OT_NETIF_THREAD
OT_NETIF_BACKBONE

}

Defines the OpenThread network interface identifiers.

Typedefs

typedef bool(* otUdpHandler)(void *aContext, const otMessage *aMessage, const otMessageInfo *aMessageInfo)

This callback allows OpenThread to provide specific handlers for certain UDP messages.

typedef struct otUdpReceiver otUdpReceiver

Represents a UDP receiver.

typedef void(* otUdpReceive)(void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo)

This callback allows OpenThread to inform the application of a received UDP message.

typedef struct otUdpSocket otUdpSocket

Represents a UDP socket.

typedef enum otNetifIdentifier otNetifIdentifier

Defines the OpenThread network interface identifiers.

Functions

otError otUdpAddReceiver(otInstance *aInstance, otUdpReceiver *aUdpReceiver)

Adds a UDP receiver.

otError otUdpRemoveReceiver(otInstance *aInstance, otUdpReceiver *aUdpReceiver)

Removes a UDP receiver.

otError otUdpSendDatagram(otInstance *aInstance, otMessage *aMessage, otMessageInfo *aMessageInfo)

Sends a UDP message without socket.

otMessage * otUdpNewMessage(otInstance *aInstance, const otMessageSettings *aSettings)

Allocate a new message buffer for sending a UDP message.

otError	otUdpOpen(otInstance *aInstance, otUdpSocket *aSocket, otUdpReceive aCallback, void *aContext) Open a UDP/IPv6 socket.
bool	otUdpIsOpen(otInstance *aInstance, const otUdpSocket *aSocket) Check if a UDP socket is open.
otError	otUdpClose(otInstance *aInstance, otUdpSocket *aSocket) Close a UDP/IPv6 socket.
otError	otUdpBind(otInstance *aInstance, otUdpSocket *aSocket, const otSockAddr *aSockName, otNetifIdentifier aNetif) Bind a UDP/IPv6 socket.
otError	otUdpConnect(otInstance *aInstance, otUdpSocket *aSocket, const otSockAddr *aSockName) Connect a UDP/IPv6 socket.
otError	otUdpSend(otInstance *aInstance, otUdpSocket *aSocket, otMessage *aMessage, const otMessageInfo *aMessageInfo) Send a UDP/IPv6 message.
otUdpSocket *	otUdpGetSockets(otInstance *aInstance) Gets the head of linked list of UDP Sockets.

Enumeration Documentation

otNetifIdentifier

```
otNetifIdentifier
```

Defines the OpenThread network interface identifiers.

	Enumerator
OT_NETIF_UNSPECIFIED	Unspecified network interface.
OT_NETIF_THREAD	The Thread interface.
OT_NETIF_BACKBONE	The Backbone interface.

Definition at line 138 of file include/openthread/udp.h

Typedef Documentation

otUdpHandler

```
typedef bool(* otUdpHandler) (void *aContext, const otMessage *aMessage, const otMessageInfo *aMessageInfo) )
(void *aContext, const otMessage *aMessage, const otMessageInfo *aMessageInfo)
```

This callback allows OpenThread to provide specific handlers for certain UDP messages.

Definition at line 63 of file include/openthread/udp.h

otUdpReceiver

```
typedef struct otUdpReceiver otUdpReceiver
```

Represents a UDP receiver.

Definition at line 74 of file include/openthread/udp.h

otUdpReceive

```
typedef void(* otUdpReceive) (void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo) )(void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo)
```

This callback allows OpenThread to inform the application of a received UDP message.

Definition at line `118` of file `include/openthread/udp.h`

otUdpSocket

```
typedef struct otUdpSocket otUdpSocket
```

Represents a UDP socket.

Definition at line `132` of file `include/openthread/udp.h`

otNetifIdentifier

```
typedef enum otNetifIdentifier otNetifIdentifier
```

Defines the OpenThread network interface identifiers.

Definition at line `143` of file `include/openthread/udp.h`

Function Documentation

otUdpAddReceiver

```
otError otUdpAddReceiver (otInstance *aInstance, otUdpReceiver *aUdpReceiver)
```

Adds a UDP receiver.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aUdpReceiver	A pointer to the UDP receiver.

Definition at line `86` of file `include/openthread/udp.h`

otUdpRemoveReceiver

```
otError otUdpRemoveReceiver (otInstance *aInstance, otUdpReceiver *aUdpReceiver)
```

Removes a UDP receiver.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aUdpReceiver	A pointer to the UDP receiver.

Definition at line 98 of file include/openthread/udp.h

otUdpSendDatagram

> otError otUdpSendDatagram (otInstance *aInstance, otMessage *aMessage, otMessageInfo *aMessageInfo)

Sends a UDP message without socket.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to a message without UDP header.
[in]	aMessageInfo	A pointer to a message info associated with aMessage.

Definition at line 112 of file include/openthread/udp.h

otUdpNewMessage

> otMessage * otUdpNewMessage (otInstance *aInstance, const otMessageSettings *aSettings)

Allocate a new message buffer for sending a UDP message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSettings	A pointer to the message settings or NULL to use default settings.

Note

- If aSettings is 'NULL', the link layer security is enabled and the message priority is set to OT_MESSAGE_PRIORITY_NORMAL by default.

Returns

- A pointer to the message buffer or NULL if no message buffers are available or parameters are invalid.

See Also

- otMessageFree

Definition at line 159 of file include/openthread/udp.h

otUdpOpen

> otError otUdpOpen (otInstance *aInstance, otUdpSocket *aSocket, otUdpReceive aCallback, void *aContext)

Open a UDP/IPv6 socket.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSocket	A pointer to a UDP socket structure.
[in]	aCallback	A pointer to the application callback function.
[in]	aContext	A pointer to application-specific context.

Definition at line 173 of file include/openthread/udp.h

otUdpIsOpen

> bool otUdpIsOpen (otInstance *aInstance, const otUdpSocket *aSocket)

Check if a UDP socket is open.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSocket	A pointer to a UDP socket structure.

Returns

- Whether the UDP socket is open.

Definition at line `184` of file `include/openthread/udp.h`

otUdpClose

> otError otUdpClose (otInstance *aInstance, otUdpSocket *aSocket)

Close a UDP/IPv6 socket.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSocket	A pointer to a UDP socket structure.

Definition at line `196` of file `include/openthread/udp.h`

otUdpBind

> otError otUdpBind (otInstance *aInstance, otUdpSocket *aSocket, const otSockAddr *aSockName, otNetifIdentifier aNetif)

Bind a UDP/IPv6 socket.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSocket	A pointer to a UDP socket structure.
[in]	aSockName	A pointer to an IPv6 socket address structure.
[in]	aNetif	The network interface to bind.

Definition at line `210` of file `include/openthread/udp.h`

otUdpConnect

> otError otUdpConnect (otInstance *aInstance, otUdpSocket *aSocket, const otSockAddr *aSockName)

Connect a UDP/IPv6 socket.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSocket	A pointer to a UDP socket structure.
[in]	aSockName	A pointer to an IPv6 socket address structure.

Definition at line `223` of file `include/openthread/udp.h`

otUdpSend

> otError otUdpSend (otInstance *aInstance, otUdpSocket *aSocket, otMessage *aMessage, const otMessageInfo *aMessageInfo)

Send a UDP/IPv6 message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSocket	A pointer to a UDP socket structure.
[in]	aMessage	A pointer to a message buffer.
[in]	aMessageInfo	A pointer to a message info structure.

If the return value is OT_ERROR_NONE, OpenThread takes ownership of `aMessage`, and the caller should no longer reference `aMessage`. If the return value is not OT_ERROR_NONE, the caller retains ownership of `aMessage`, including freeing `aMessage` if the message buffer is no longer needed.

Definition at line `242` of file `include/openthread/udp.h`

otUdpGetSockets

> otUdpSocket * otUdpGetSockets (otInstance *aInstance)

Gets the head of linked list of UDP Sockets.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the head of UDP Socket linked list.

Definition at line `252` of file `include/openthread/udp.h`

otUdpReceiver

Represents a UDP receiver.

Public Attributes

struct otUdpReceiver *	mNext A pointer to the next UDP receiver (internal use only).
otUdpHandler	mHandler A function pointer to the receiver callback.
void *	mContext A pointer to application-specific context.

Public Attribute Documentation

mNext

```
struct otUdpReceiver* otUdpReceiver::mNext
```

A pointer to the next UDP receiver (internal use only).

Definition at line 71 of file include/openthread/udp.h

mHandler

```
otUdpHandler otUdpReceiver::mHandler
```

A function pointer to the receiver callback.

Definition at line 72 of file include/openthread/udp.h

mContext

```
void* otUdpReceiver::mContext
```

A pointer to application-specific context.

Definition at line 73 of file include/openthread/udp.h

otUdpSocket

Represents a UDP socket.

Public Attributes

otSockAddr	mSockName The local IPv6 socket address.
otSockAddr	mPeerName The peer IPv6 socket address.
otUdpReceive	mHandler A function pointer to the application callback.
void *	mContext A pointer to application-specific context.
void *	mHandle A handle to platform's UDP.
struct otUdpSocket *	mNext A pointer to the next UDP socket (internal use only).

Public Attribute Documentation

mSockName

```
otSockAddr otUdpSocket::mSockName
```

The local IPv6 socket address.

Definition at line 126 of file include/openthread/udp.h

mPeerName

```
otSockAddr otUdpSocket::mPeerName
```

The peer IPv6 socket address.

Definition at line 127 of file include/openthread/udp.h

mHandler

```
otUdpReceive otUdpSocket::mHandler
```

A function pointer to the application callback.

Definition at line 128 of file include/openthread/udp.h

> void* otUdpSocket::mContext

A pointer to application-specific context.

Definition at line `129` of file `include/openthread/udp.h`

mHandle

> void* otUdpSocket::mHandle

A handle to platform's UDP.

Definition at line `130` of file `include/openthread/udp.h`

mNext

> struct otUdpSocket* otUdpSocket::mNext

A pointer to the next UDP socket (internal use only).

Definition at line `131` of file `include/openthread/udp.h`

UDP Forward

UDP Forward

This module includes functions for UDP forward feature.

The functions in this module are available when udp-forward feature (`OPENTHREAD_CONFIG_UDP_FORWARD_ENABLE`) is enabled.

Typedefs

typedef void(* otUdpForwarder)(otMessage *aMessage, uint16_t aPeerPort, otIp6Address *aPeerAddr, uint16_t aSockPort, void *aContext)
Pointer delivers the UDP packet to host and host should send the packet through its own network stack.

Functions

void otUdpForwardSetForwarder(otInstance *aInstance, otUdpForwarder aForwarder, void *aContext)
Set UDP forward callback to deliver UDP packets to host.

void otUdpForwardReceive(otInstance *aInstance, otMessage *aMessage, uint16_t aPeerPort, const otIp6Address *aPeerAddr, uint16_t aSockPort)
Handle a UDP packet received from host.

bool otUdpIsPortInUse(otInstance *aInstance, uint16_t port)
Determines if the given UDP port is exclusively opened by OpenThread API.

Typedef Documentation

otUdpForwarder

```
typedef void(* otUdpForwarder) (otMessage *aMessage, uint16_t aPeerPort, otIp6Address *aPeerAddr, uint16_t aSockPort, void *aContext) )(otMessage *aMessage, uint16_t aPeerPort, otIp6Address *aPeerAddr, uint16_t aSockPort, void *aContext)
```

Pointer delivers the UDP packet to host and host should send the packet through its own network stack.

Parameters

[in]	aMessage	A pointer to the UDP Message.
[in]	aPeerPort	The destination UDP port.
[in]	aPeerAddr	A pointer to the destination IPv6 address.
[in]	aSockPort	The source UDP port.
[in]	aContext	A pointer to application-specific context.

Definition at line `282` of file `include/openthread/udp.h`

Function Documentation

otUdpForwardSetForwarder

> void otUdpForwardSetForwarder (otInstance *aInstance, otUdpForwarder aForwarder, void *aContext)

Set UDP forward callback to deliver UDP packets to host.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aForwarder	A pointer to a function called to forward UDP packet to host.
[in]	aContext	A pointer to application-specific context.

Definition at line `296` of file `include/openthread/udp.h`

otUdpForwardReceive

> void otUdpForwardReceive (otInstance *aInstance, otMessage *aMessage, uint16_t aPeerPort, const otIp6Address *aPeerAddr, uint16_t aSockPort)

Handle a UDP packet received from host.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the UDP Message.
[in]	aPeerPort	The source UDP port.
[in]	aPeerAddr	A pointer to the source address.
[in]	aSockPort	The destination UDP port.

Warnings

- No matter the call success or fail, the message is freed.

Definition at line `310` of file `include/openthread/udp.h`

otUdpIsPortInUse

> bool otUdpIsPortInUse (otInstance *aInstance, uint16_t port)

Determines if the given UDP port is exclusively opened by OpenThread API.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	port	UDP port number to verify.

Definition at line `326` of file `include/openthread/udp.h`

Link

Link

Modules

Link

Link Metrics

Raw Link

Link

Link

This module includes functions that control link-layer configuration.

Modules

otThreadLinkInfo

otMacFilterEntry

otMacCounters

otActiveScanResult

otEnergyScanResult

Enumerations

enum otMacFilterAddressMode {

OT_MAC_FILTER_ADDRESS_MODE_DISABLED
OT_MAC_FILTER_ADDRESS_MODE_ALLOWLIST
OT_MAC_FILTER_ADDRESS_MODE_DENYLIST

}

Defines address mode of the mac filter.

Typedefs

typedef struct otThreadLinkInfo — otThreadLinkInfo
Represents link-specific information for messages received from the Thread radio.

typedef uint8_t — otMacFilterIterator
Used to iterate through mac filter entries.

typedef enum otMacFilterAddressMode — otMacFilterAddressMode
Defines address mode of the mac filter.

typedef struct otMacFilterEntry — otMacFilterEntry
Represents a Mac Filter entry.

typedef struct otMacCounters — otMacCounters
Represents the MAC layer counters.

typedef struct otActiveScanResult — otActiveScanResult
Represents a received IEEE 802.15.4 Beacon.

typedef struct otEnergyScanResult — otEnergyScanResult
Represents an energy scan result.

typedef void(* otHandleActiveScanResult)(otActiveScanResult *aResult, void *aContext)
Pointer is called during an IEEE 802.15.4 Active Scan when an IEEE 802.15.4 Beacon is received or the scan completes.

typedef void(* otHandleEnergyScanResult)(otEnergyScanResult *aResult, void *aContext)
Pointer is called during an IEEE 802.15.4 Energy Scan when the result for a channel is ready or the scan completes.

typedef void(* otLinkPcapCallback)(const otRadioFrame *aFrame, bool aIsTx, void *aContext)
Pointer is called when an IEEE 802.15.4 frame is received.

Functions

otError otLinkActiveScan(otInstance *aInstance, uint32_t aScanChannels, uint16_t aScanDuration, otHandleActiveScanResult aCallback, void *aCallbackContext)
Starts an IEEE 802.15.4 Active Scan.

bool otLinkIsActiveScanInProgress(otInstance *aInstance)
Indicates whether or not an IEEE 802.15.4 Active Scan is currently in progress.

otError otLinkEnergyScan(otInstance *aInstance, uint32_t aScanChannels, uint16_t aScanDuration, otHandleEnergyScanResult aCallback, void *aCallbackContext)
Starts an IEEE 802.15.4 Energy Scan.

bool otLinkIsEnergyScanInProgress(otInstance *aInstance)
Indicates whether or not an IEEE 802.15.4 Energy Scan is currently in progress.

otError otLinkSendDataRequest(otInstance *aInstance)
Enqueues an IEEE 802.15.4 Data Request message for transmission.

bool otLinkIsInTransmitState(otInstance *aInstance)
Indicates whether or not an IEEE 802.15.4 MAC is in the transmit state.

uint8_t otLinkGetChannel(otInstance *aInstance)
Get the IEEE 802.15.4 channel.

otError otLinkSetChannel(otInstance *aInstance, uint8_t aChannel)
Set the IEEE 802.15.4 channel.

uint32_t otLinkGetSupportedChannelMask(otInstance *aInstance)
Get the supported channel mask of MAC layer.

otError otLinkSetSupportedChannelMask(otInstance *aInstance, uint32_t aChannelMask)
Set the supported channel mask of MAC layer.

const otExtAddress * otLinkGetExtendedAddress(otInstance *aInstance)
Gets the IEEE 802.15.4 Extended Address.

otError otLinkSetExtendedAddress(otInstance *aInstance, const otExtAddress *aExtAddress)
Sets the IEEE 802.15.4 Extended Address.

void otLinkGetFactoryAssignedIeeeEui64(otInstance *aInstance, otExtAddress *aEui64)
Get the factory-assigned IEEE EUI-64.

otPanId otLinkGetPanId(otInstance *aInstance)
Get the IEEE 802.15.4 PAN ID.

otError otLinkSetPanId(otInstance *aInstance, otPanId aPanId)
Set the IEEE 802.15.4 PAN ID.

uint32_t otLinkGetPollPeriod(otInstance *aInstance)
Get the data poll period of sleepy end device.

otError otLinkSetPollPeriod(otInstance *aInstance, uint32_t aPollPeriod)
Set/clear user-specified/external data poll period for sleepy end device.

otShortAddress	otLinkGetShortAddress(otInstance *aInstance) Get the IEEE 802.15.4 Short Address.
uint8_t	otLinkGetMaxFrameRetriesDirect(otInstance *aInstance) Returns the maximum number of frame retries during direct transmission.
void	otLinkSetMaxFrameRetriesDirect(otInstance *aInstance, uint8_t aMaxFrameRetriesDirect) Sets the maximum number of frame retries during direct transmission.
uint8_t	otLinkGetMaxFrameRetriesIndirect(otInstance *aInstance) Returns the maximum number of frame retries during indirect transmission.
void	otLinkSetMaxFrameRetriesIndirect(otInstance *aInstance, uint8_t aMaxFrameRetriesIndirect) Sets the maximum number of frame retries during indirect transmission.
otMacFilterAddressMode	otLinkFilterGetAddressMode(otInstance *aInstance) Gets the address mode of MAC filter.
void	otLinkFilterSetAddressMode(otInstance *aInstance, otMacFilterAddressMode aMode) Sets the address mode of MAC filter.
otError	otLinkFilterAddAddress(otInstance *aInstance, const otExtAddress *aExtAddress) Adds an Extended Address to MAC filter.
void	otLinkFilterRemoveAddress(otInstance *aInstance, const otExtAddress *aExtAddress) Removes an Extended Address from MAC filter.
void	otLinkFilterClearAddresses(otInstance *aInstance) Clears all the Extended Addresses from MAC filter.
otError	otLinkFilterGetNextAddress(otInstance *aInstance, otMacFilterIterator *aIterator, otMacFilterEntry *aEntry) Gets an in-use address filter entry.
otError	otLinkFilterAddRssIn(otInstance *aInstance, const otExtAddress *aExtAddress, int8_t aRss) Adds the specified Extended Address to the `RssIn` list (or modifies an existing address in the `RssIn` list) and sets the received signal strength (in dBm) entry for messages from that address.
void	otLinkFilterRemoveRssIn(otInstance *aInstance, const otExtAddress *aExtAddress) Removes the specified Extended Address from the `RssIn` list.
void	otLinkFilterSetDefaultRssIn(otInstance *aInstance, int8_t aRss) Sets the default received signal strength (in dBm) on MAC Filter.
void	otLinkFilterClearDefaultRssIn(otInstance *aInstance) Clears any previously set default received signal strength (in dBm) on MAC Filter.
void	otLinkFilterClearAllRssIn(otInstance *aInstance) Clears all the received signal strength (`rss`) and link quality indicator (`lqi`) entries (including defaults) from the `RssIn` list.
otError	otLinkFilterGetNextRssIn(otInstance *aInstance, otMacFilterIterator *aIterator, otMacFilterEntry *aEntry) Gets an in-use RssIn filter entry.
void	otLinkSetRadioFilterEnabled(otInstance *aInstance, bool aFilterEnabled) Enables/disables IEEE 802.15.4 radio filter mode.
bool	otLinkIsRadioFilterEnabled(otInstance *aInstance) Indicates whether the IEEE 802.15.4 radio filter is enabled or not.
uint8_t	otLinkConvertRssToLinkQuality(otInstance *aInstance, int8_t aRss) Converts received signal strength to link quality.

int8_t	otLinkConvertLinkQualityToRss(otInstance *aInstance, uint8_t aLinkQuality) Converts link quality to typical received signal strength.
const uint32_t *	otLinkGetTxDirectRetrySuccessHistogram(otInstance *aInstance, uint8_t *aNumberOfEntries) Gets histogram of retries for a single direct packet until success.
const uint32_t *	otLinkGetTxIndirectRetrySuccessHistogram(otInstance *aInstance, uint8_t *aNumberOfEntries) Gets histogram of retries for a single indirect packet until success.
void	otLinkResetTxRetrySuccessHistogram(otInstance *aInstance) Clears histogram statistics for direct and indirect transmissions.
const otMacCounters *	otLinkGetCounters(otInstance *aInstance) Get the MAC layer counters.
void	otLinkResetCounters(otInstance *aInstance) Resets the MAC layer counters.
void	otLinkSetPcapCallback(otInstance *aInstance, otLinkPcapCallback aPcapCallback, void *aCallbackContext) Registers a callback to provide received raw IEEE 802.15.4 frames.
bool	otLinkIsPromiscuous(otInstance *aInstance) Indicates whether or not promiscuous mode is enabled at the link layer.
otError	otLinkSetPromiscuous(otInstance *aInstance, bool aPromiscuous) Enables or disables the link layer promiscuous mode.
uint8_t	otLinkGetCslChannel(otInstance *aInstance) Gets the CSL channel.
otError	otLinkSetCslChannel(otInstance *aInstance, uint8_t aChannel) Sets the CSL channel.
uint32_t	otLinkGetCslPeriod(otInstance *aInstance) Gets the CSL period in microseconds.
otError	otLinkSetCslPeriod(otInstance *aInstance, uint32_t aPeriod) Sets the CSL period in microseconds.
uint32_t	otLinkGetCslTimeout(otInstance *aInstance) Gets the CSL timeout.
otError	otLinkSetCslTimeout(otInstance *aInstance, uint32_t aTimeout) Sets the CSL timeout in seconds.
uint16_t	otLinkGetCcaFailureRate(otInstance *aInstance) Returns the current CCA (Clear Channel Assessment) failure rate.
otError	otLinkSetEnabled(otInstance *aInstance, bool aEnable) Enables or disables the link layer.
bool	otLinkIsEnabled(otInstance *aInstance) Indicates whether or not the link layer is enabled.
bool	otLinkIsCslEnabled(otInstance *aInstance) Indicates whether or not CSL is enabled.
bool	otLinkIsCslSupported(otInstance *aInstance) Indicates whether the device is connected to a parent which supports CSL.
otError	otLinkSendEmptyData(otInstance *aInstance) Instructs the device to send an empty IEEE 802.15.4 data frame.

otError otLinkSetRegion(otInstance *aInstance, uint16_t aRegionCode)
Sets the region code.

otError otLinkGetRegion(otInstance *aInstance, uint16_t *aRegionCode)
Get the region code.

Macros

#define OT_US_PER_TEN_SYMBOLS OT_RADIO_TEN_SYMBOLS_TIME
Time for 10 symbols in units of microseconds.

#define OT_MAC_FILTER_FIXED_RSS_DISABLED 127
Used to indicate no fixed received signal strength was set.

#define OT_MAC_FILTER_ITERATOR_INIT 0
Initializer for otMacFilterIterator.

#define OT_LINK_CSL_PERIOD_TEN_SYMBOLS_UNIT_IN_USEC (160)
Represents CSL period ten symbols unit in microseconds.

Enumeration Documentation

otMacFilterAddressMode

> otMacFilterAddressMode

Defines address mode of the mac filter.

Enumerator	
OT_MAC_FILTER_ADDRESS_MODE_DISABLED	Address filter is disabled.
OT_MAC_FILTER_ADDRESS_MODE_ALLOWLIST	Allowlist address filter mode is enabled.
OT_MAC_FILTER_ADDRESS_MODE_DENYLIST	Denylist address filter mode is enabled.

Definition at line `92` of file `include/openthread/link.h`

Typedef Documentation

otThreadLinkInfo

> typedef struct otThreadLinkInfo otThreadLinkInfo

Represents link-specific information for messages received from the Thread radio.

Definition at line `76` of file `include/openthread/link.h`

otMacFilterIterator

> typedef uint8_t otMacFilterIterator

Used to iterate through mac filter entries.

Definition at line `86` of file `include/openthread/link.h`

otMacFilterAddressMode

```
typedef enum otMacFilterAddressMode otMacFilterAddressMode
```

Defines address mode of the mac filter.

Definition at line 97 of file include/openthread/link.h

otMacFilterEntry

```
typedef struct otMacFilterEntry otMacFilterEntry
```

Represents a Mac Filter entry.

Definition at line 107 of file include/openthread/link.h

otMacCounters

```
typedef struct otMacCounters otMacCounters
```

Represents the MAC layer counters.

Definition at line 374 of file include/openthread/link.h

otActiveScanResult

```
typedef struct otActiveScanResult otActiveScanResult
```

Represents a received IEEE 802.15.4 Beacon.

Definition at line 398 of file include/openthread/link.h

otEnergyScanResult

```
typedef struct otEnergyScanResult otEnergyScanResult
```

Represents an energy scan result.

Definition at line 408 of file include/openthread/link.h

otHandleActiveScanResult

```
typedef void(* otHandleActiveScanResult) (otActiveScanResult *aResult, void *aContext) )(otActiveScanResult *aResult, void *aContext)
```

Pointer is called during an IEEE 802.15.4 Active Scan when an IEEE 802.15.4 Beacon is received or the scan completes.

Parameters

[in]	aResult	A valid pointer to the beacon information or NULL when the active scan completes.

[in]	aContext	A pointer to application-specific context.

Definition at line `418` of file `include/openthread/link.h`

otHandleEnergyScanResult

```
typedef void(* otHandleEnergyScanResult) (otEnergyScanResult *aResult, void *aContext) )(otEnergyScanResult *aResult, void *aContext)
```

Pointer is called during an IEEE 802.15.4 Energy Scan when the result for a channel is ready or the scan completes.

Parameters

[in]	aResult	A valid pointer to the energy scan result information or NULL when the energy scan completes.
[in]	aContext	A pointer to application-specific context.

Definition at line `456` of file `include/openthread/link.h`

otLinkPcapCallback

```
typedef void(* otLinkPcapCallback) (const otRadioFrame *aFrame, bool aIsTx, void *aContext) )(const otRadioFrame *aFrame, bool aIsTx, void *aContext)
```

Pointer is called when an IEEE 802.15.4 frame is received.

Parameters

[in]	aFrame	A pointer to the received IEEE 802.15.4 frame.
[in]	aIsTx	Whether this frame is transmitted, not received.
[in]	aContext	A pointer to application-specific context.

Note

- This callback is called after FCS processing and `aFrame` may not contain the actual FCS that was received.
- This callback is called before IEEE 802.15.4 security processing.

Definition at line `996` of file `include/openthread/link.h`

Function Documentation

otLinkActiveScan

```
otError otLinkActiveScan (otInstance *aInstance, uint32_t aScanChannels, uint16_t aScanDuration, otHandleActiveScanResult aCallback, void *aCallbackContext)
```

Starts an IEEE 802.15.4 Active Scan.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aScanChannels	A bit vector indicating which channels to scan (e.g. OT_CHANNEL_11_MASK).
[in]	aScanDuration	The time in milliseconds to spend scanning each channel.
[in]	aCallback	A pointer to a function called on receiving a beacon or scan completes.
[in]	aCallbackContext	A pointer to application-specific context.

Definition at line 433 of file include/openthread/link.h

otLinkIsActiveScanInProgress

```
bool otLinkIsActiveScanInProgress (otInstance *aInstance)
```

Indicates whether or not an IEEE 802.15.4 Active Scan is currently in progress.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- true if an IEEE 802.15.4 Active Scan is in progress, false otherwise.

Definition at line 446 of file include/openthread/link.h

otLinkEnergyScan

```
otError otLinkEnergyScan (otInstance *aInstance, uint32_t aScanChannels, uint16_t aScanDuration,
otHandleEnergyScanResult aCallback, void *aCallbackContext)
```

Starts an IEEE 802.15.4 Energy Scan.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aScanChannels	A bit vector indicating on which channels to perform energy scan.
[in]	aScanDuration	The time in milliseconds to spend scanning each channel.
[in]	aCallback	A pointer to a function called to pass on scan result on indicate scan completion.
[in]	aCallbackContext	A pointer to application-specific context.

Definition at line 471 of file include/openthread/link.h

otLinkIsEnergyScanInProgress

```
bool otLinkIsEnergyScanInProgress (otInstance *aInstance)
```

Indicates whether or not an IEEE 802.15.4 Energy Scan is currently in progress.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- true if an IEEE 802.15.4 Energy Scan is in progress, false otherwise.

Definition at line 485 of file include/openthread/link.h

otLinkSendDataRequest

```
otError otLinkSendDataRequest (otInstance *aInstance)
```

Enqueues an IEEE 802.15.4 Data Request message for transmission.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `497` of file `include/openthread/link.h`

otLinkIsInTransmitState

```
bool otLinkIsInTransmitState (otInstance *aInstance)
```

Indicates whether or not an IEEE 802.15.4 MAC is in the transmit state.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

MAC module is in the transmit state during CSMA/CA procedure, CCA, Data, Beacon or Data Request frame transmission and receiving an ACK of a transmitted frame. MAC module is not in the transmit state during transmission of an ACK frame or a Beacon Request frame.

Returns

- true if an IEEE 802.15.4 MAC is in the transmit state, false otherwise.

Definition at line `511` of file `include/openthread/link.h`

otLinkGetChannel

```
uint8_t otLinkGetChannel (otInstance *aInstance)
```

Get the IEEE 802.15.4 channel.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The IEEE 802.15.4 channel.

See Also

- otLinkSetChannel

Definition at line `523` of file `include/openthread/link.h`

otLinkSetChannel

```
otError otLinkSetChannel (otInstance *aInstance, uint8_t aChannel)
```

Set the IEEE 802.15.4 channel.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChannel	The IEEE 802.15.4 channel.

Succeeds only when Thread protocols are disabled. A successful call to this function invalidates the Active and Pending Operational Datasets in non-volatile memory.

See Also

- otLinkGetChannel

Definition at line 541 of file include/openthread/link.h

otLinkGetSupportedChannelMask

```
uint32_t otLinkGetSupportedChannelMask (otInstance *aInstance)
```

Get the supported channel mask of MAC layer.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The supported channel mask as uint32_t with bit 0 (lsb) mapping to channel 0, bit 1 to channel 1, so on.

Definition at line 551 of file include/openthread/link.h

otLinkSetSupportedChannelMask

```
otError otLinkSetSupportedChannelMask (otInstance *aInstance, uint32_t aChannelMask)
```

Set the supported channel mask of MAC layer.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChannelMask	The supported channel mask (bit 0 or lsb mapping to channel 0, and so on).

Succeeds only when Thread protocols are disabled.

Definition at line 565 of file include/openthread/link.h

otLinkGetExtendedAddress

```
const otExtAddress * otLinkGetExtendedAddress (otInstance *aInstance)
```

Gets the IEEE 802.15.4 Extended Address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the IEEE 802.15.4 Extended Address.

Definition at line 575 of file include/openthread/link.h

otLinkSetExtendedAddress

> otError otLinkSetExtendedAddress (otInstance *aInstance, const otExtAddress *aExtAddress)

Sets the IEEE 802.15.4 Extended Address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	A pointer to the IEEE 802.15.4 Extended Address.

Note

- Only succeeds when Thread protocols are disabled.

Definition at line `590` of file `include/openthread/link.h`

otLinkGetFactoryAssignedIeeeEui64

> void otLinkGetFactoryAssignedIeeeEui64 (otInstance *aInstance, otExtAddress *aEui64)

Get the factory-assigned IEEE EUI-64.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[out]	aEui64	A pointer to where the factory-assigned IEEE EUI-64 is placed.

Definition at line `599` of file `include/openthread/link.h`

otLinkGetPanId

> otPanId otLinkGetPanId (otInstance *aInstance)

Get the IEEE 802.15.4 PAN ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The IEEE 802.15.4 PAN ID.

See Also

- otLinkSetPanId

Definition at line `611` of file `include/openthread/link.h`

otLinkSetPanId

> otError otLinkSetPanId (otInstance *aInstance, otPanId aPanId)

Set the IEEE 802.15.4 PAN ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPanId	The IEEE 802.15.4 PAN ID.

Succeeds only when Thread protocols are disabled. A successful call to this function also invalidates the Active and Pending Operational Datasets in non-volatile memory.

See Also

- otLinkGetPanId

Definition at line 629 of file include/openthread/link.h

otLinkGetPollPeriod

```
uint32_t otLinkGetPollPeriod (otInstance *aInstance)
```

Get the data poll period of sleepy end device.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The data poll period of sleepy end device in milliseconds.

See Also

- otLinkSetPollPeriod

Definition at line 641 of file include/openthread/link.h

otLinkSetPollPeriod

```
otError otLinkSetPollPeriod (otInstance *aInstance, uint32_t aPollPeriod)
```

Set/clear user-specified/external data poll period for sleepy end device.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPollPeriod	data poll period in milliseconds.

Note

- This function updates only poll period of sleepy end device. To update child timeout the function otThreadSetChildTimeout() shall be called.
- Minimal non-zero value should be OPENTHREAD_CONFIG_MAC_MINIMUM_POLL_PERIOD (10ms). Or zero to clear user-specified poll period.
- User-specified value should be no more than the maximal value 0x3FFFFFF ((1 << 26) - 1) allowed, otherwise it would be clipped by the maximal value.

See Also

- otLinkGetPollPeriod

Definition at line 664 of file include/openthread/link.h

otLinkGetShortAddress

> otShortAddress otLinkGetShortAddress (otInstance *aInstance)

Get the IEEE 802.15.4 Short Address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the IEEE 802.15.4 Short Address.

Definition at line `674` of file `include/openthread/link.h`

otLinkGetMaxFrameRetriesDirect

> uint8_t otLinkGetMaxFrameRetriesDirect (otInstance *aInstance)

Returns the maximum number of frame retries during direct transmission.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The maximum number of retries during direct transmission.

Definition at line `684` of file `include/openthread/link.h`

otLinkSetMaxFrameRetriesDirect

> void otLinkSetMaxFrameRetriesDirect (otInstance *aInstance, uint8_t aMaxFrameRetriesDirect)

Sets the maximum number of frame retries during direct transmission.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMaxFrameRetriesDirect	The maximum number of retries during direct transmission.

Definition at line `693` of file `include/openthread/link.h`

otLinkGetMaxFrameRetriesIndirect

> uint8_t otLinkGetMaxFrameRetriesIndirect (otInstance *aInstance)

Returns the maximum number of frame retries during indirect transmission.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The maximum number of retries during indirect transmission.

Definition at line 703 of file include/openthread/link.h

otLinkSetMaxFrameRetriesIndirect

```
void otLinkSetMaxFrameRetriesIndirect (otInstance *aInstance, uint8_t aMaxFrameRetriesIndirect)
```

Sets the maximum number of frame retries during indirect transmission.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMaxFrameRetriesIndirect	The maximum number of retries during indirect transmission.

Definition at line 712 of file include/openthread/link.h

otLinkFilterGetAddressMode

```
otMacFilterAddressMode otLinkFilterGetAddressMode (otInstance *aInstance)
```

Gets the address mode of MAC filter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Is available when OPENTHREAD_CONFIG_MAC_FILTER_ENABLE configuration is enabled.

Returns

- the address mode.

Definition at line 724 of file include/openthread/link.h

otLinkFilterSetAddressMode

```
void otLinkFilterSetAddressMode (otInstance *aInstance, otMacFilterAddressMode aMode)
```

Sets the address mode of MAC filter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMode	The address mode to set.

Is available when OPENTHREAD_CONFIG_MAC_FILTER_ENABLE configuration is enabled.

Definition at line 735 of file include/openthread/link.h

otLinkFilterAddAddress

```
otError otLinkFilterAddAddress (otInstance *aInstance, const otExtAddress *aExtAddress)
```

Adds an Extended Address to MAC filter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	A pointer to the Extended Address (MUST NOT be NULL).

Is available when `OPENTHREAD_CONFIG_MAC_FILTER_ENABLE` configuration is enabled.

Definition at line `749` of file `include/openthread/link.h`

otLinkFilterRemoveAddress

```
void otLinkFilterRemoveAddress (otInstance *aInstance, const otExtAddress *aExtAddress)
```

Removes an Extended Address from MAC filter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	A pointer to the Extended Address (MUST NOT be NULL).

Is available when `OPENTHREAD_CONFIG_MAC_FILTER_ENABLE` configuration is enabled.

No action is performed if there is no existing entry in Filter matching the given Extended Address.

Definition at line `762` of file `include/openthread/link.h`

otLinkFilterClearAddresses

```
void otLinkFilterClearAddresses (otInstance *aInstance)
```

Clears all the Extended Addresses from MAC filter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Is available when `OPENTHREAD_CONFIG_MAC_FILTER_ENABLE` configuration is enabled.

Definition at line `772` of file `include/openthread/link.h`

otLinkFilterGetNextAddress

```
otError otLinkFilterGetNextAddress (otInstance *aInstance, otMacFilterIterator *aIterator, otMacFilterEntry *aEntry)
```

Gets an in-use address filter entry.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the MAC filter iterator context. To get the first in-use address filter entry, it should be set to OT_MAC_FILTER_ITERATOR_INIT. MUST NOT be NULL.
[out]	aEntry	A pointer to where the information is placed. MUST NOT be NULL.

Is available when `OPENTHREAD_CONFIG_MAC_FILTER_ENABLE` configuration is enabled.

Definition at line `788` of file `include/openthread/link.h`

otLinkFilterAddRssIn

> otError otLinkFilterAddRssIn (otInstance *aInstance, const otExtAddress *aExtAddress, int8_t aRss)

Adds the specified Extended Address to the `RssIn` list (or modifies an existing address in the `RssIn` list) and sets the received signal strength (in dBm) entry for messages from that address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	A pointer to the IEEE 802.15.4 Extended Address. MUST NOT be NULL.
[in]	aRss	A received signal strength (in dBm).

The Extended Address does not necessarily have to be in the `address allowlist/denylist` filter to set the `rss`. **Note**

- The `RssIn` list contains Extended Addresses whose `rss` or link quality indicator (`lqi`) values have been set to be different from the defaults.

Is available when `OPENTHREAD_CONFIG_MAC_FILTER_ENABLE` configuration is enabled.

Definition at line `808` of file `include/openthread/link.h`

otLinkFilterRemoveRssIn

> void otLinkFilterRemoveRssIn (otInstance *aInstance, const otExtAddress *aExtAddress)

Removes the specified Extended Address from the `RssIn` list.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	A pointer to the IEEE 802.15.4 Extended Address. MUST NOT be NULL.

Once removed from the `RssIn` list, this MAC address will instead use the default `rss` and `lqi` settings, assuming defaults have been set. (If no defaults have been set, the over-air signal is used.)

Is available when `OPENTHREAD_CONFIG_MAC_FILTER_ENABLE` configuration is enabled.

No action is performed if there is no existing entry in the `RssIn` list matching the specified Extended Address.

Definition at line `824` of file `include/openthread/link.h`

otLinkFilterSetDefaultRssIn

> void otLinkFilterSetDefaultRssIn (otInstance *aInstance, int8_t aRss)

Sets the default received signal strength (in dBm) on MAC Filter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aRss	The default received signal strength (in dBm) to set.

Is available when `OPENTHREAD_CONFIG_MAC_FILTER_ENABLE` configuration is enabled.

The default RSS value is used for all received frames from addresses for which there is no explicit RSS-IN entry in the Filter list (added using `otLinkFilterAddRssIn()`).

Definition at line 838 of file include/openthread/link.h

otLinkFilterClearDefaultRssIn

```
void otLinkFilterClearDefaultRssIn (otInstance *aInstance)
```

Clears any previously set default received signal strength (in dBm) on MAC Filter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Is available when OPENTHREAD_CONFIG_MAC_FILTER_ENABLE configuration is enabled.

Definition at line 848 of file include/openthread/link.h

otLinkFilterClearAllRssIn

```
void otLinkFilterClearAllRssIn (otInstance *aInstance)
```

Clears all the received signal strength (rss) and link quality indicator (lqi) entries (including defaults) from the RssIn list.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Performing this action means that all Extended Addresses will use the on-air signal.

Is available when OPENTHREAD_CONFIG_MAC_FILTER_ENABLE configuration is enabled.

Definition at line 860 of file include/openthread/link.h

otLinkFilterGetNextRssIn

```
otError otLinkFilterGetNextRssIn (otInstance *aInstance, otMacFilterIterator *aIterator, otMacFilterEntry *aEntry)
```

Gets an in-use RssIn filter entry.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the MAC filter iterator context. MUST NOT be NULL. To get the first entry, it should be set to OT_MAC_FILTER_ITERATOR_INIT.
[out]	aEntry	A pointer to where the information is placed. The last entry would have the extended address as all 0xff to indicate the default received signal strength if it was set. aEntry MUST NOT be NULL.

Is available when OPENTHREAD_CONFIG_MAC_FILTER_ENABLE configuration is enabled.

Definition at line 878 of file include/openthread/link.h

otLinkSetRadioFilterEnabled

```
void otLinkSetRadioFilterEnabled (otInstance *aInstance, bool aFilterEnabled)
```

Enables/disables IEEE 802.15.4 radio filter mode.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aFilterEnabled	TRUE to enable radio filter, FALSE to disable

Is available when `OPENTHREAD_CONFIG_MAC_FILTER_ENABLE` configuration is enabled.

The radio filter is mainly intended for testing. It can be used to temporarily block all tx/rx on the 802.15.4 radio. When radio filter is enabled, radio is put to sleep instead of receive (to ensure device does not receive any frame and/or potentially send ack). Also the frame transmission requests return immediately without sending the frame over the air (return "no ack" error if ack is requested, otherwise return success).

Definition at line `894` of file `include/openthread/link.h`

otLinkIsRadioFilterEnabled

```
bool otLinkIsRadioFilterEnabled (otInstance *aInstance)
```

Indicates whether the IEEE 802.15.4 radio filter is enabled or not.

Parameters

N/A	aInstance	

Is available when `OPENTHREAD_CONFIG_MAC_FILTER_ENABLE` configuration is enabled.

Definition at line `905` of file `include/openthread/link.h`

otLinkConvertRssToLinkQuality

```
uint8_t otLinkConvertRssToLinkQuality (otInstance *aInstance, int8_t aRss)
```

Converts received signal strength to link quality.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aRss	The received signal strength value to be converted.

Returns

- Link quality value mapping to `aRss`.

Definition at line `916` of file `include/openthread/link.h`

otLinkConvertLinkQualityToRss

```
int8_t otLinkConvertLinkQualityToRss (otInstance *aInstance, uint8_t aLinkQuality)
```

Converts link quality to typical received signal strength.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aLinkQuality	LinkQuality value, should be in range [0,3].

Returns

- Typical platform received signal strength mapping to `aLinkQuality`.

Definition at line `927` of file `include/openthread/link.h`

otLinkGetTxDirectRetrySuccessHistogram

```
const uint32_t * otLinkGetTxDirectRetrySuccessHistogram (otInstance *aInstance, uint8_t *aNumberOfEntries)
```

Gets histogram of retries for a single direct packet until success.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aNumberOfEntries	A pointer to where the size of returned histogram array is placed.

Is valid when OPENTHREAD_CONFIG_MAC_RETRY_SUCCESS_HISTOGRAM_ENABLE configuration is enabled.

Returns

- A pointer to the histogram of retries (in a form of an array). The n-th element indicates that the packet has been sent with n-th retry.

Definition at line `940` of file `include/openthread/link.h`

otLinkGetTxIndirectRetrySuccessHistogram

```
const uint32_t * otLinkGetTxIndirectRetrySuccessHistogram (otInstance *aInstance, uint8_t *aNumberOfEntries)
```

Gets histogram of retries for a single indirect packet until success.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aNumberOfEntries	A pointer to where the size of returned histogram array is placed.

Is valid when OPENTHREAD_CONFIG_MAC_RETRY_SUCCESS_HISTOGRAM_ENABLE configuration is enabled.

Returns

- A pointer to the histogram of retries (in a form of an array). The n-th element indicates that the packet has been sent with n-th retry.

Definition at line `954` of file `include/openthread/link.h`

otLinkResetTxRetrySuccessHistogram

```
void otLinkResetTxRetrySuccessHistogram (otInstance *aInstance)
```

Clears histogram statistics for direct and indirect transmissions.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Is valid when OPENTHREAD_CONFIG_MAC_RETRY_SUCCESS_HISTOGRAM_ENABLE configuration is enabled.

Definition at line `964` of file `include/openthread/link.h`

otLinkGetCounters

```
const otMacCounters * otLinkGetCounters (otInstance *aInstance)
```

Get the MAC layer counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the MAC layer counters.

Definition at line `974` of file `include/openthread/link.h`

otLinkResetCounters

```
void otLinkResetCounters (otInstance *aInstance)
```

Resets the MAC layer counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `982` of file `include/openthread/link.h`

otLinkSetPcapCallback

```
void otLinkSetPcapCallback (otInstance *aInstance, otLinkPcapCallback aPcapCallback, void *aCallbackContext)
```

Registers a callback to provide received raw IEEE 802.15.4 frames.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPcapCallback	A pointer to a function that is called when receiving an IEEE 802.15.4 link frame or NULL to disable the callback.
[in]	aCallbackContext	A pointer to application-specific context.

Definition at line `1007` of file `include/openthread/link.h`

otLinkIsPromiscuous

```
bool otLinkIsPromiscuous (otInstance *aInstance)
```

Indicates whether or not promiscuous mode is enabled at the link layer.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `1018` of file `include/openthread/link.h`

otLinkSetPromiscuous

```
otError otLinkSetPromiscuous (otInstance *aInstance, bool aPromiscuous)
```

Enables or disables the link layer promiscuous mode.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPromiscuous	true to enable promiscuous mode, or false otherwise.

Note

- Promiscuous mode may only be enabled when the Thread interface is disabled.

Definition at line `1033` of file `include/openthread/link.h`

otLinkGetCslChannel

```
uint8_t otLinkGetCslChannel (otInstance *aInstance)
```

Gets the CSL channel.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The CSL channel.

Definition at line `1043` of file `include/openthread/link.h`

otLinkSetCslChannel

```
otError otLinkSetCslChannel (otInstance *aInstance, uint8_t aChannel)
```

Sets the CSL channel.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChannel	The CSL sample channel. Channel value should be `0` (Set CSL Channel unspecified) or within the range [1, 10] (if 915-MHz supported) and [11, 26] (if 2.4 GHz supported).

Definition at line `1056` of file `include/openthread/link.h`

otLinkGetCslPeriod

```
uint32_t otLinkGetCslPeriod (otInstance *aInstance)
```

Gets the CSL period in microseconds.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The CSL period in microseconds.

Definition at line `1074` of file `include/openthread/link.h`

otLinkSetCslPeriod

```
otError otLinkSetCslPeriod (otInstance *aInstance, uint32_t aPeriod)
```

Sets the CSL period in microseconds.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPeriod	The CSL period in microseconds.

Disable CSL by setting this parameter to `0`.

The CSL period MUST be a multiple of `OT_LINK_CSL_PERIOD_TEN_SYMBOLS_UNIT_IN_USEC`, otherwise `OT_ERROR_INVALID_ARGS` is returned.

Definition at line `1089` of file `include/openthread/link.h`

otLinkGetCslTimeout

```
uint32_t otLinkGetCslTimeout (otInstance *aInstance)
```

Gets the CSL timeout.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The CSL timeout in seconds.

Definition at line `1099` of file `include/openthread/link.h`

otLinkSetCslTimeout

```
otError otLinkSetCslTimeout (otInstance *aInstance, uint32_t aTimeout)
```

Sets the CSL timeout in seconds.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aTimeout	The CSL timeout in seconds.

Definition at line `1111` of file `include/openthread/link.h`

otLinkGetCcaFailureRate

> uint16_t otLinkGetCcaFailureRate (otInstance *aInstance)

Returns the current CCA (Clear Channel Assessment) failure rate.

Parameters

N/A	aInstance	

The rate is maintained over a window of (roughly) last `OPENTHREAD_CONFIG_CCA_FAILURE_RATE_AVERAGING_WINDOW` frame transmissions.

Returns

- The CCA failure rate with maximum value `0xffff` corresponding to 100% failure rate.

Definition at line `1122` of file `include/openthread/link.h`

otLinkSetEnabled

> otError otLinkSetEnabled (otInstance *aInstance, bool aEnable)

Enables or disables the link layer.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnable	true to enable the link layer, or false otherwise.

Note

- The link layer may only be enabled / disabled when the Thread Interface is disabled.

Definition at line `1137` of file `include/openthread/link.h`

otLinkIsEnabled

> bool otLinkIsEnabled (otInstance *aInstance)

Indicates whether or not the link layer is enabled.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `1148` of file `include/openthread/link.h`

otLinkIsCslEnabled

> bool otLinkIsCslEnabled (otInstance *aInstance)

Indicates whether or not CSL is enabled.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 1159 of file include/openthread/link.h

otLinkIsCslSupported

```
bool otLinkIsCslSupported (otInstance *aInstance)
```

Indicates whether the device is connected to a parent which supports CSL.

Parameters

N/A	aInstance	

Definition at line 1168 of file include/openthread/link.h

otLinkSendEmptyData

```
otError otLinkSendEmptyData (otInstance *aInstance)
```

Instructs the device to send an empty IEEE 802.15.4 data frame.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Is only supported on an Rx-Off-When-Idle device to send an empty data frame to its parent. Note: available only when OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE is enabled.

Definition at line 1183 of file include/openthread/link.h

otLinkSetRegion

```
otError otLinkSetRegion (otInstance *aInstance, uint16_t aRegionCode)
```

Sets the region code.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aRegionCode	The radio region code. The aRegionCode >> 8 is first ascii char and the aRegionCode & 0xff is the second ascii char.

The radio region format is the 2-bytes ascii representation of the ISO 3166 alpha-2 code.

Definition at line 1199 of file include/openthread/link.h

otLinkGetRegion

```
otError otLinkGetRegion (otInstance *aInstance, uint16_t *aRegionCode)
```

Get the region code.

Parameters

[in]	aInstance	The OpenThread instance structure.

[out]	aRegionCode	The radio region code. The `aRegionCode >> 8` is first ascii char and the `aRegionCode & 0xff` is the second ascii char.

The radio region format is the 2-bytes ascii representation of the ISO 3166 alpha-2 code.

Definition at line `1216` of file `include/openthread/link.h`

Macro Definition Documentation

OT_US_PER_TEN_SYMBOLS

```
#define OT_US_PER_TEN_SYMBOLS
```

Value:

```
OT_RADIO_TEN_SYMBOLS_TIME
```

Time for 10 symbols in units of microseconds.

Definition at line `55` of file `include/openthread/link.h`

OT_MAC_FILTER_FIXED_RSS_DISABLED

```
#define OT_MAC_FILTER_FIXED_RSS_DISABLED
```

Value:

```
127
```

Used to indicate no fixed received signal strength was set.

Definition at line `82` of file `include/openthread/link.h`

OT_MAC_FILTER_ITERATOR_INIT

```
#define OT_MAC_FILTER_ITERATOR_INIT
```

Value:

```
0
```

Initializer for otMacFilterIterator.

Definition at line `84` of file `include/openthread/link.h`

OT_LINK_CSL_PERIOD_TEN_SYMBOLS_UNIT_IN_USEC

```
#define OT_LINK_CSL_PERIOD_TEN_SYMBOLS_UNIT_IN_USEC
```

Value:

```
(160)
```

Represents CSL period ten symbols unit in microseconds.

The CSL period (in micro seconds) MUST be a multiple of this value.

Definition at line 1064 of file include/openthread/link.h

otThreadLinkInfo

Represents link-specific information for messages received from the Thread radio.

Public Attributes

uint16_t	mPanId Source PAN ID.
uint8_t	mChannel 802.15.4 Channel
int8_t	mRss Received Signal Strength in dBm.
uint8_t	mLqi Link Quality Indicator for a received message.
bool	mLinkSecurity Indicates whether or not link security is enabled.
bool	mIsDstPanIdBroadcast Indicates whether or not destination PAN ID is broadcast.
uint8_t	mTimeSyncSeq The time sync sequence.
int64_t	mNetworkTimeOffset The time offset to the Thread network time, in microseconds.
uint8_t	mRadioType Radio link type.

Public Attribute Documentation

mPanId

```
uint16_t otThreadLinkInfo::mPanId
```

Source PAN ID.

Definition at line `63` of file `include/openthread/link.h`

mChannel

```
uint8_t otThreadLinkInfo::mChannel
```

802.15.4 Channel

Definition at line `64` of file `include/openthread/link.h`

mRss

> int8_t otThreadLinkInfo::mRss

Received Signal Strength in dBm.

Definition at line `65` of file `include/openthread/link.h`

mLqi

> uint8_t otThreadLinkInfo::mLqi

Link Quality Indicator for a received message.

Definition at line `66` of file `include/openthread/link.h`

mLinkSecurity

> bool otThreadLinkInfo::mLinkSecurity

Indicates whether or not link security is enabled.

Definition at line `67` of file `include/openthread/link.h`

mIsDstPanIdBroadcast

> bool otThreadLinkInfo::mIsDstPanIdBroadcast

Indicates whether or not destination PAN ID is broadcast.

Definition at line `68` of file `include/openthread/link.h`

mTimeSyncSeq

> uint8_t otThreadLinkInfo::mTimeSyncSeq

The time sync sequence.

Definition at line `71` of file `include/openthread/link.h`

mNetworkTimeOffset

> int64_t otThreadLinkInfo::mNetworkTimeOffset

The time offset to the Thread network time, in microseconds.

Definition at line `72` of file `include/openthread/link.h`

mRadioType

uint8_t otThreadLinkInfo::mRadioType

Radio link type.

Definition at line 75 of file include/openthread/link.h

otMacFilterEntry

Represents a Mac Filter entry.

Public Attributes

otExtAddress	mExtAddress IEEE 802.15.4 Extended Address.
int8_t	mRssIn Received signal strength.

Public Attribute Documentation

mExtAddress

```
otExtAddress otMacFilterEntry::mExtAddress
```

IEEE 802.15.4 Extended Address.

Definition at line 105 of file include/openthread/link.h

mRssIn

```
int8_t otMacFilterEntry::mRssIn
```

Received signal strength.

Definition at line 106 of file include/openthread/link.h

otMacCounters

Represents the MAC layer counters.

Public Attributes

uint32_t **mTxTotal**
The total number of unique MAC frame transmission requests.

uint32_t **mTxUnicast**
The total number of unique unicast MAC frame transmission requests.

uint32_t **mTxBroadcast**
The total number of unique broadcast MAC frame transmission requests.

uint32_t **mTxAckRequested**
The total number of unique MAC frame transmission requests with requested acknowledgment.

uint32_t **mTxAcked**
The total number of unique MAC frame transmission requests that were acked.

uint32_t **mTxNoAckRequested**
The total number of unique MAC frame transmission requests without requested acknowledgment.

uint32_t **mTxData**
The total number of unique MAC Data frame transmission requests.

uint32_t **mTxDataPoll**
The total number of unique MAC Data Poll frame transmission requests.

uint32_t **mTxBeacon**
The total number of unique MAC Beacon frame transmission requests.

uint32_t **mTxBeaconRequest**
The total number of unique MAC Beacon Request frame transmission requests.

uint32_t **mTxOther**
The total number of unique other MAC frame transmission requests.

uint32_t **mTxRetry**
The total number of MAC retransmission attempts.

uint32_t **mTxDirectMaxRetryExpiry**
The total number of unique MAC transmission packets that meet maximal retry limit for direct packets.

uint32_t **mTxIndirectMaxRetryExpiry**
The total number of unique MAC transmission packets that meet maximal retry limit for indirect packets.

uint32_t **mTxErrCca**
The total number of CCA failures.

uint32_t **mTxErrAbort**
The total number of unique MAC transmission request failures cause by an abort error.

uint32_t **mTxErrBusyChannel**
The total number of unique MAC transmission requests failures caused by a busy channel (a CSMA/CA fail).

uint32_t mRxTotal
The total number of received frames.

uint32_t mRxUnicast
The total number of unicast frames received.

uint32_t mRxBroadcast
The total number of broadcast frames received.

uint32_t mRxData
The total number of MAC Data frames received.

uint32_t mRxDataPoll
The total number of MAC Data Poll frames received.

uint32_t mRxBeacon
The total number of MAC Beacon frames received.

uint32_t mRxBeaconRequest
The total number of MAC Beacon Request frames received.

uint32_t mRxOther
The total number of other types of frames received.

uint32_t mRxAddressFiltered
The total number of frames dropped by MAC Filter module, for example received from denylisted node.

uint32_t mRxDestAddrFiltered
The total number of frames dropped by destination address check, for example received frame for other node.

uint32_t mRxDuplicated
The total number of frames dropped due to duplication, that is when the frame has been already received.

uint32_t mRxErrNoFrame
The total number of frames dropped because of missing or malformed content.

uint32_t mRxErrUnknownNeighbor
The total number of frames dropped due to unknown neighbor.

uint32_t mRxErrInvalidSrcAddr
The total number of frames dropped due to invalid source address.

uint32_t mRxErrSec
The total number of frames dropped due to security error.

uint32_t mRxErrFcs
The total number of frames dropped due to invalid FCS.

uint32_t mRxErrOther
The total number of frames dropped due to other error.

Public Attribute Documentation

mTxTotal

> uint32_t otMacCounters::mTxTotal

The total number of unique MAC frame transmission requests.

Note that this counter is incremented for each MAC transmission request only by one, regardless of the amount of CCA failures, CSMA-CA attempts, or retransmissions.

This increment rule applies to the following counters:

- mTxUnicast
- mTxBroadcast
- mTxAckRequested
- mTxNoAckRequested
- mTxData
- mTxDataPoll
- mTxBeacon
- mTxBeaconRequest
- mTxOther
- mTxErrAbort
- mTxErrBusyChannel

The following equations are valid:

- mTxTotal = mTxUnicast + mTxBroadcast
- mTxTotal = mTxAckRequested + mTxNoAckRequested
- mTxTotal = mTxData + mTxDataPoll + mTxBeacon + mTxBeaconRequest + mTxOther

Definition at line 140 of file include/openthread/link.h

mTxUnicast

```
uint32_t otMacCounters::mTxUnicast
```

The total number of unique unicast MAC frame transmission requests.

Definition at line 146 of file include/openthread/link.h

mTxBroadcast

```
uint32_t otMacCounters::mTxBroadcast
```

The total number of unique broadcast MAC frame transmission requests.

Definition at line 152 of file include/openthread/link.h

mTxAckRequested

```
uint32_t otMacCounters::mTxAckRequested
```

The total number of unique MAC frame transmission requests with requested acknowledgment.

Definition at line 158 of file include/openthread/link.h

mTxAcked

```
uint32_t otMacCounters::mTxAcked
```

The total number of unique MAC frame transmission requests that were acked.

Definition at line 164 of file include/openthread/link.h

mTxNoAckRequested

```
uint32_t otMacCounters::mTxNoAckRequested
```

The total number of unique MAC frame transmission requests without requested acknowledgment.

Definition at line 170 of file include/openthread/link.h

mTxData

```
uint32_t otMacCounters::mTxData
```

The total number of unique MAC Data frame transmission requests.

Definition at line 176 of file include/openthread/link.h

mTxDataPoll

```
uint32_t otMacCounters::mTxDataPoll
```

The total number of unique MAC Data Poll frame transmission requests.

Definition at line 182 of file include/openthread/link.h

mTxBeacon

```
uint32_t otMacCounters::mTxBeacon
```

The total number of unique MAC Beacon frame transmission requests.

Definition at line 188 of file include/openthread/link.h

mTxBeaconRequest

```
uint32_t otMacCounters::mTxBeaconRequest
```

The total number of unique MAC Beacon Request frame transmission requests.

Definition at line 194 of file include/openthread/link.h

mTxOther

```
uint32_t otMacCounters::mTxOther
```

The total number of unique other MAC frame transmission requests.

This counter is currently used for counting out-of-band frames.

Definition at line 202 of file include/openthread/link.h

mTxRetry

```
uint32_t otMacCounters::mTxRetry
```

The total number of MAC retransmission attempts.

Note that this counter is incremented by one for each retransmission attempt that may be triggered by lack of acknowledgement, CSMA/CA failure, or other type of transmission error. The mTxRetry counter is incremented both for unicast and broadcast MAC frames.

Modify the following configuration parameters to control the amount of retransmissions in the system:

- OPENTHREAD_CONFIG_MAC_DEFAULT_MAX_FRAME_RETRIES_DIRECT
- OPENTHREAD_CONFIG_MAC_DEFAULT_MAX_FRAME_RETRIES_INDIRECT
- OPENTHREAD_CONFIG_MAC_TX_NUM_BCAST
- OPENTHREAD_CONFIG_MAC_MAX_CSMA_BACKOFFS_DIRECT
- OPENTHREAD_CONFIG_MAC_MAX_CSMA_BACKOFFS_INDIRECT

Currently, this counter is invalid if the platform's radio driver capability includes OT_RADIO_CAPS_TRANSMIT_RETRIES.

Definition at line 223 of file include/openthread/link.h

mTxDirectMaxRetryExpiry

```
uint32_t otMacCounters::mTxDirectMaxRetryExpiry
```

The total number of unique MAC transmission packets that meet maximal retry limit for direct packets.

Definition at line 229 of file include/openthread/link.h

mTxIndirectMaxRetryExpiry

```
uint32_t otMacCounters::mTxIndirectMaxRetryExpiry
```

The total number of unique MAC transmission packets that meet maximal retry limit for indirect packets.

Definition at line 235 of file include/openthread/link.h

mTxErrCca

```
uint32_t otMacCounters::mTxErrCca
```

The total number of CCA failures.

The meaning of this counter can be different and it depends on the platform's radio driver capabilities.

If OT_RADIO_CAPS_CSMA_BACKOFF is enabled, this counter represents the total number of full CSMA/CA failed attempts and it is incremented by one also for each retransmission (in case of a CSMA/CA fail).

If OT_RADIO_CAPS_TRANSMIT_RETRIES is enabled, this counter represents the total number of full CSMA/CA failed attempts and it is incremented by one for each individual data frame request (regardless of the amount of retransmissions).

Definition at line 250 of file include/openthread/link.h

mTxErrAbort

```
uint32_t otMacCounters::mTxErrAbort
```

The total number of unique MAC transmission request failures cause by an abort error.

Definition at line 256 of file include/openthread/link.h

mTxErrBusyChannel

```
uint32_t otMacCounters::mTxErrBusyChannel
```

The total number of unique MAC transmission requests failures caused by a busy channel (a CSMA/CA fail).

Definition at line 262 of file include/openthread/link.h

mRxTotal

```
uint32_t otMacCounters::mRxTotal
```

The total number of received frames.

This counter counts all frames reported by the platform's radio driver, including frames that were dropped, for example because of an FCS error.

Definition at line 271 of file include/openthread/link.h

mRxUnicast

```
uint32_t otMacCounters::mRxUnicast
```

The total number of unicast frames received.

Definition at line 277 of file include/openthread/link.h

mRxBroadcast

```
uint32_t otMacCounters::mRxBroadcast
```

The total number of broadcast frames received.

Definition at line 283 of file include/openthread/link.h

mRxData

uint32_t otMacCounters::mRxData

The total number of MAC Data frames received.

Definition at line 289 of file include/openthread/link.h

mRxDataPoll

uint32_t otMacCounters::mRxDataPoll

The total number of MAC Data Poll frames received.

Definition at line 295 of file include/openthread/link.h

mRxBeacon

uint32_t otMacCounters::mRxBeacon

The total number of MAC Beacon frames received.

Definition at line 301 of file include/openthread/link.h

mRxBeaconRequest

uint32_t otMacCounters::mRxBeaconRequest

The total number of MAC Beacon Request frames received.

Definition at line 307 of file include/openthread/link.h

mRxOther

uint32_t otMacCounters::mRxOther

The total number of other types of frames received.

Definition at line 313 of file include/openthread/link.h

mRxAddressFiltered

uint32_t otMacCounters::mRxAddressFiltered

The total number of frames dropped by MAC Filter module, for example received from denylisted node.

Definition at line 319 of file include/openthread/link.h

mRxDestAddrFiltered

uint32_t otMacCounters::mRxDestAddrFiltered

The total number of frames dropped by destination address check, for example received frame for other node.

Definition at line 325 of file include/openthread/link.h

mRxDuplicated

uint32_t otMacCounters::mRxDuplicated

The total number of frames dropped due to duplication, that is when the frame has been already received.

This counter may be incremented, for example when ACK frame generated by the receiver hasn't reached transmitter node which performed retransmission.

Definition at line 334 of file include/openthread/link.h

mRxErrNoFrame

uint32_t otMacCounters::mRxErrNoFrame

The total number of frames dropped because of missing or malformed content.

Definition at line 340 of file include/openthread/link.h

mRxErrUnknownNeighbor

uint32_t otMacCounters::mRxErrUnknownNeighbor

The total number of frames dropped due to unknown neighbor.

Definition at line 346 of file include/openthread/link.h

mRxErrInvalidSrcAddr

uint32_t otMacCounters::mRxErrInvalidSrcAddr

The total number of frames dropped due to invalid source address.

Definition at line 352 of file include/openthread/link.h

mRxErrSec

uint32_t otMacCounters::mRxErrSec

The total number of frames dropped due to security error.

This counter may be incremented, for example when lower than expected Frame Counter is used to encrypt the frame.

Definition at line 361 of file include/openthread/link.h

mRxErrFcs

```
uint32_t otMacCounters::mRxErrFcs
```

The total number of frames dropped due to invalid FCS.

Definition at line 367 of file include/openthread/link.h

mRxErrOther

```
uint32_t otMacCounters::mRxErrOther
```

The total number of frames dropped due to other error.

Definition at line 373 of file include/openthread/link.h

otActiveScanResult

Represents a received IEEE 802.15.4 Beacon.

Public Attributes

otExtAddress	mExtAddress IEEE 802.15.4 Extended Address.
otNetworkName	mNetworkName Thread Network Name.
otExtendedPanId	mExtendedPanId Thread Extended PAN ID.
otSteeringData	mSteeringData Steering Data.
uint16_t	mPanId IEEE 802.15.4 PAN ID.
uint16_t	mJoinerUdpPort Joiner UDP Port.
uint8_t	mChannel IEEE 802.15.4 Channel.
int8_t	mRssi RSSI (dBm)
uint8_t	mLqi LQI.
unsigned int	mVersion Version.
bool	mIsNative Native Commissioner flag.
bool	mDiscover Result from MLE Discovery.
bool	mIsJoinable Joining Permitted flag.

Public Attribute Documentation

mExtAddress

> otExtAddress otActiveScanResult::mExtAddress

IEEE 802.15.4 Extended Address.

Definition at line `382` of file `include/openthread/link.h`

mNetworkName

```
otNetworkName otActiveScanResult::mNetworkName
```

Thread Network Name.

Definition at line `383` of file `include/openthread/link.h`

mExtendedPanId

```
otExtendedPanId otActiveScanResult::mExtendedPanId
```

Thread Extended PAN ID.

Definition at line `384` of file `include/openthread/link.h`

mSteeringData

```
otSteeringData otActiveScanResult::mSteeringData
```

Steering Data.

Definition at line `385` of file `include/openthread/link.h`

mPanId

```
uint16_t otActiveScanResult::mPanId
```

IEEE 802.15.4 PAN ID.

Definition at line `386` of file `include/openthread/link.h`

mJoinerUdpPort

```
uint16_t otActiveScanResult::mJoinerUdpPort
```

Joiner UDP Port.

Definition at line `387` of file `include/openthread/link.h`

mChannel

```
uint8_t otActiveScanResult::mChannel
```

IEEE 802.15.4 Channel.

Definition at line `388` of file `include/openthread/link.h`

mRssi

```
int8_t otActiveScanResult::mRssi
```

RSSI (dBm)

Definition at line `389` of file `include/openthread/link.h`

mLqi

```
uint8_t otActiveScanResult::mLqi
```

LQI.

Definition at line `390` of file `include/openthread/link.h`

mVersion

```
unsigned int otActiveScanResult::mVersion
```

Version.

Definition at line `391` of file `include/openthread/link.h`

mIsNative

```
bool otActiveScanResult::mIsNative
```

Native Commissioner flag.

Definition at line `392` of file `include/openthread/link.h`

mDiscover

```
bool otActiveScanResult::mDiscover
```

Result from MLE Discovery.

Definition at line `393` of file `include/openthread/link.h`

mIsJoinable

```
bool otActiveScanResult::mIsJoinable
```

Joining Permitted flag.

Definition at line `397` of file `include/openthread/link.h`

otEnergyScanResult

Represents an energy scan result.

Public Attributes

uint8_t	mChannel IEEE 802.15.4 Channel.
int8_t	mMaxRssi The max RSSI (dBm)

Public Attribute Documentation

mChannel

```
uint8_t otEnergyScanResult::mChannel
```

IEEE 802.15.4 Channel.

Definition at line 406 of file include/openthread/link.h

mMaxRssi

```
int8_t otEnergyScanResult::mMaxRssi
```

The max RSSI (dBm)

Definition at line 407 of file include/openthread/link.h

Link Metrics

Link Metrics

This module includes functions that control the Link Metrics protocol.

Modules

otLinkMetricsValues

otLinkMetricsSeriesFlags

Enumerations

enum otLinkMetricsEnhAckFlags {

OT_LINK_METRICS_ENH_ACK_CLEAR = 0
OT_LINK_METRICS_ENH_ACK_REGISTER = 1

}

Enhanced-ACK Flags.

enum otLinkMetricsStatus {

OT_LINK_METRICS_STATUS_SUCCESS = 0
OT_LINK_METRICS_STATUS_CANNOT_SUPPORT_NEW_SERIES = 1
OT_LINK_METRICS_STATUS_SERIESID_ALREADY_REGISTERED = 2
OT_LINK_METRICS_STATUS_SERIESID_NOT_RECOGNIZED = 3
OT_LINK_METRICS_STATUS_NO_MATCHING_FRAMES_RECEIVED = 4
OT_LINK_METRICS_STATUS_OTHER_ERROR = 254

}

Link Metrics Status values.

Typedefs

typedef struct otLinkMetricsValues — otLinkMetricsValues
Represents the result (value) for a Link Metrics query.

typedef struct otLinkMetricsSeriesFlags — otLinkMetricsSeriesFlags
Represents which frames are accounted in a Forward Tracking Series.

typedef enum otLinkMetricsEnhAckFlags — otLinkMetricsEnhAckFlags
Enhanced-ACK Flags.

typedef enum otLinkMetricsStatus — otLinkMetricsStatus
Link Metrics Status values.

typedef void(* otLinkMetricsReportCallback)(const otIp6Address *aSource, const otLinkMetricsValues *aMetricsValues, otLinkMetricsStatus aStatus, void *aContext)
Pointer is called when a Link Metrics report is received.

typedef void(* otLinkMetricsMgmtResponseCallback)(const otIp6Address *aSource, otLinkMetricsStatus aStatus, void *aContext)

Pointer is called when a Link Metrics Management Response is received.

typedef void(* otLinkMetricsEnhAckProbingIeReportCallback)(otShortAddress aShortAddress, const otExtAddress *aExtAddress, const otLinkMetricsValues *aMetricsValues, void *aContext)

Pointer is called when Enh-ACK Probing IE is received.

Functions

otError otLinkMetricsQuery(otInstance *aInstance, const otIp6Address *aDestination, uint8_t aSeriesId, const otLinkMetrics *aLinkMetricsFlags, otLinkMetricsReportCallback aCallback, void *aCallbackContext)

Sends an MLE Data Request to query Link Metrics.

otError otLinkMetricsConfigForwardTrackingSeries(otInstance *aInstance, const otIp6Address *aDestination, uint8_t aSeriesId, otLinkMetricsSeriesFlags aSeriesFlags, const otLinkMetrics *aLinkMetricsFlags, otLinkMetricsMgmtResponseCallback aCallback, void *aCallbackContext)

Sends an MLE Link Metrics Management Request to configure or clear a Forward Tracking Series.

otError otLinkMetricsConfigEnhAckProbing(otInstance *aInstance, const otIp6Address *aDestination, otLinkMetricsEnhAckFlags aEnhAckFlags, const otLinkMetrics *aLinkMetricsFlags, otLinkMetricsMgmtResponseCallback aCallback, void *aCallbackContext, otLinkMetricsEnhAckProbingIeReportCallback aEnhAckCallback, void *aEnhAckCallbackContext)

Sends an MLE Link Metrics Management Request to configure/clear an Enhanced-ACK Based Probing.

otError otLinkMetricsSendLinkProbe(otInstance *aInstance, const otIp6Address *aDestination, uint8_t aSeriesId, uint8_t aLength)

Sends an MLE Link Probe message.

void otLinkMetricsManagerSetEnabled(otInstance *aInstance, bool aEnable)

Enable or disable Link Metrics Manager.

otError otLinkMetricsManagerGetMetricsValueByExtAddr(otInstance *aInstance, const otExtAddress *aExtAddress, otLinkMetricsValues *aLinkMetricsValues)

Get Link Metrics data of a neighbor by its extended address.

Enumeration Documentation

otLinkMetricsEnhAckFlags

> otLinkMetricsEnhAckFlags

Enhanced-ACK Flags.

These are used in Enhanced-ACK Based Probing to indicate whether to register or clear the probing.

Enumerator	
OT_LINK_METRICS_ENH_ACK_CLEAR	Clear.
OT_LINK_METRICS_ENH_ACK_REGISTER	Register.

Definition at line 88 of file include/openthread/link_metrics.h

otLinkMetricsStatus

> otLinkMetricsStatus

Link Metrics Status values.

Enumerator	
OT_LINK_METRICS_STATUS_SUCCESS	
OT_LINK_METRICS_STATUS_CANNOT_SUPPORT_NEW_SERIES	
OT_LINK_METRICS_STATUS_SERIESID_ALREADY_REGISTERED	
OT_LINK_METRICS_STATUS_SERIESID_NOT_RECOGNIZED	
OT_LINK_METRICS_STATUS_NO_MATCHING_FRAMES_RECEIVED	
OT_LINK_METRICS_STATUS_OTHER_ERROR	

Definition at line `98` of file `include/openthread/link_metrics.h`

Typedef Documentation

otLinkMetricsValues

```
typedef struct otLinkMetricsValues otLinkMetricsValues
```

Represents the result (value) for a Link Metrics query.

Definition at line `68` of file `include/openthread/link_metrics.h`

otLinkMetricsSeriesFlags

```
typedef struct otLinkMetricsSeriesFlags otLinkMetricsSeriesFlags
```

Represents which frames are accounted in a Forward Tracking Series.

Definition at line `80` of file `include/openthread/link_metrics.h`

otLinkMetricsEnhAckFlags

```
typedef enum otLinkMetricsEnhAckFlags otLinkMetricsEnhAckFlags
```

Enhanced-ACK Flags.

These are used in Enhanced-ACK Based Probing to indicate whether to register or clear the probing.

Definition at line `92` of file `include/openthread/link_metrics.h`

otLinkMetricsStatus

```
typedef enum otLinkMetricsStatus otLinkMetricsStatus
```

Link Metrics Status values.

Definition at line `106` of file `include/openthread/link_metrics.h`

otLinkMetricsReportCallback

```
typedef void(* otLinkMetricsReportCallback) (const otIp6Address *aSource, const otLinkMetricsValues *aMetricsValues,
otLinkMetricsStatus aStatus, void *aContext) )(const otIp6Address *aSource, const otLinkMetricsValues *aMetricsValues,
otLinkMetricsStatus aStatus, void *aContext)
```

Pointer is called when a Link Metrics report is received.

Parameters

[in]	aSource	A pointer to the source address.
[in]	aMetricsValues	A pointer to the Link Metrics values (the query result).
[in]	aStatus	The status code in the report (only useful when `aMetricsValues` is NULL).
[in]	aContext	A pointer to application-specific context.

Definition at line `117` of file `include/openthread/link_metrics.h`

otLinkMetricsMgmtResponseCallback

```
typedef void(* otLinkMetricsMgmtResponseCallback) (const otIp6Address *aSource, otLinkMetricsStatus aStatus, void *aContext) )(const otIp6Address *aSource, otLinkMetricsStatus aStatus, void *aContext)
```

Pointer is called when a Link Metrics Management Response is received.

Parameters

[in]	aSource	A pointer to the source address.
[in]	aStatus	The status code in the response.
[in]	aContext	A pointer to application-specific context.

Definition at line `129` of file `include/openthread/link_metrics.h`

otLinkMetricsEnhAckProbingIeReportCallback

```
typedef void(* otLinkMetricsEnhAckProbingIeReportCallback) (otShortAddress aShortAddress, const otExtAddress *aExtAddress, const otLinkMetricsValues *aMetricsValues, void *aContext) )(otShortAddress aShortAddress, const otExtAddress *aExtAddress, const otLinkMetricsValues *aMetricsValues, void *aContext)
```

Pointer is called when Enh-ACK Probing IE is received.

Parameters

[in]	aShortAddress	The Mac short address of the Probing Subject.
[in]	aExtAddress	A pointer to the Mac extended address of the Probing Subject.
[in]	aMetricsValues	A pointer to the Link Metrics values obtained from the IE.
[in]	aContext	A pointer to application-specific context.

Definition at line `142` of file `include/openthread/link_metrics.h`

Function Documentation

otLinkMetricsQuery

```
otError otLinkMetricsQuery (otInstance *aInstance, const otIp6Address *aDestination, uint8_t aSeriesId, const otLinkMetrics *aLinkMetricsFlags, otLinkMetricsReportCallback aCallback, void *aCallbackContext)
```

Sends an MLE Data Request to query Link Metrics.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDestination	A pointer to the destination address.
[in]	aSeriesId	The Series ID to query about, 0 for Single Probe.
[in]	aLinkMetricsFlags	A pointer to flags specifying what metrics to query.
[in]	aCallback	A pointer to a function that is called when Link Metrics report is received.
[in]	aCallbackContext	A pointer to application-specific context.

It could be either Single Probe or Forward Tracking Series.

Definition at line `165` of file `include/openthread/link_metrics.h`

otLinkMetricsConfigForwardTrackingSeries

> otError otLinkMetricsConfigForwardTrackingSeries (otInstance *aInstance, const otIp6Address *aDestination, uint8_t aSeriesId, otLinkMetricsSeriesFlags aSeriesFlags, const otLinkMetrics *aLinkMetricsFlags, otLinkMetricsMgmtResponseCallback aCallback, void *aCallbackContext)

Sends an MLE Link Metrics Management Request to configure or clear a Forward Tracking Series.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDestination	A pointer to the destination address.
[in]	aSeriesId	The Series ID to operate with.
[in]	aSeriesFlags	The Series Flags that specifies which frames are to be accounted.
[in]	aLinkMetricsFlags	A pointer to flags specifying what metrics to query. Should be `NULL` when `aSeriesFlags` is `0`.
[in]	aCallback	A pointer to a function that is called when Link Metrics Management Response is received.
[in]	aCallbackContext	A pointer to application-specific context.

Definition at line `192` of file `include/openthread/link_metrics.h`

otLinkMetricsConfigEnhAckProbing

> otError otLinkMetricsConfigEnhAckProbing (otInstance *aInstance, const otIp6Address *aDestination, otLinkMetricsEnhAckFlags aEnhAckFlags, const otLinkMetrics *aLinkMetricsFlags, otLinkMetricsMgmtResponseCallback aCallback, void *aCallbackContext, otLinkMetricsEnhAckProbingleReportCallback aEnhAckCallback, void *aEnhAckCallbackContext)

Sends an MLE Link Metrics Management Request to configure/clear an Enhanced-ACK Based Probing.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDestination	A pointer to the destination address.
[in]	aEnhAckFlags	Enh-ACK Flags to indicate whether to register or clear the probing. `0` to clear and `1` to register. Other values are reserved.
[in]	aLinkMetricsFlags	A pointer to flags specifying what metrics to query. Should be `NULL` when `aEnhAckFlags` is `0`.
[in]	aCallback	A pointer to a function that is called when an Enhanced Ack with Link Metrics is received.
[in]	aCallbackContext	A pointer to application-specific context.
N/A	aEnhAckCallback	

N/A	aEnhAckCallbackContext	

This functionality requires OT_LINK_METRICS_INITIATOR feature enabled.

Definition at line 221 of file include/openthread/link_metrics.h

otLinkMetricsSendLinkProbe

> otError otLinkMetricsSendLinkProbe (otInstance *aInstance, const otIp6Address *aDestination, uint8_t aSeriesId, uint8_t aLength)

Sends an MLE Link Probe message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDestination	A pointer to the destination address.
[in]	aSeriesId	The Series ID [1, 254] which the Probe message aims at.
[in]	aLength	The length of the data payload in Link Probe TLV, [0, 64] (per Thread 1.2 spec, 4.4.37).

Definition at line 245 of file include/openthread/link_metrics.h

otLinkMetricsManagerSetEnabled

> void otLinkMetricsManagerSetEnabled (otInstance *aInstance, bool aEnable)

Enable or disable Link Metrics Manager.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnable	A boolean indicating to enable or disable.

Definition at line 257 of file include/openthread/link_metrics.h

otLinkMetricsManagerGetMetricsValueByExtAddr

> otError otLinkMetricsManagerGetMetricsValueByExtAddr (otInstance *aInstance, const otExtAddress *aExtAddress, otLinkMetricsValues *aLinkMetricsValues)

Get Link Metrics data of a neighbor by its extended address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	A pointer to the Mac extended address of the Probing Subject.
[out]	aLinkMetricsValues	A pointer to the Link Metrics values of the subject.

Definition at line 271 of file include/openthread/link_metrics.h

otLinkMetricsValues

Represents the result (value) for a Link Metrics query.

Public Attributes

otLinkMetrics	mMetrics Specifies which metrics values are present/included.
uint32_t	mPduCountValue The value of Pdu Count.
uint8_t	mLqiValue The value LQI.
uint8_t	mLinkMarginValue The value of Link Margin.
int8_t	mRssiValue The value of Rssi.

Public Attribute Documentation

mMetrics

> otLinkMetrics otLinkMetricsValues::mMetrics

Specifies which metrics values are present/included.

Definition at line `62` of file `include/openthread/link_metrics.h`

mPduCountValue

> uint32_t otLinkMetricsValues::mPduCountValue

The value of Pdu Count.

Definition at line `64` of file `include/openthread/link_metrics.h`

mLqiValue

> uint8_t otLinkMetricsValues::mLqiValue

The value LQI.

Definition at line `65` of file `include/openthread/link_metrics.h`

mLinkMarginValue

uint8_t otLinkMetricsValues::mLinkMarginValue

The value of Link Margin.

Definition at line 66 of file include/openthread/link_metrics.h

mRssiValue

int8_t otLinkMetricsValues::mRssiValue

The value of Rssi.

Definition at line 67 of file include/openthread/link_metrics.h

otLinkMetricsSeriesFlags

Represents which frames are accounted in a Forward Tracking Series.

Public Attributes

bool	mLinkProbe MLE Link Probe.
bool	mMacData MAC Data frame.
bool	mMacDataRequest MAC Data Request.
bool	mMacAck MAC Ack.

Public Attribute Documentation

mLinkProbe

```
bool otLinkMetricsSeriesFlags::mLinkProbe
```

MLE Link Probe.

Definition at line `76` of file `include/openthread/link_metrics.h`

mMacData

```
bool otLinkMetricsSeriesFlags::mMacData
```

MAC Data frame.

Definition at line `77` of file `include/openthread/link_metrics.h`

mMacDataRequest

```
bool otLinkMetricsSeriesFlags::mMacDataRequest
```

MAC Data Request.

Definition at line `78` of file `include/openthread/link_metrics.h`

mMacAck

```
bool otLinkMetricsSeriesFlags::mMacAck
```

MAC Ack.

Definition at line 79 of file include/openthread/link_metrics.h

Raw Link

Raw Link

This module includes functions that control the raw link-layer configuration.

Typedefs

typedef void(* otLinkRawReceiveDone)(otInstance *aInstance, otRadioFrame *aFrame, otError aError)
Pointer on receipt of a IEEE 802.15.4 frame.

typedef void(* otLinkRawTransmitDone)(otInstance *aInstance, otRadioFrame *aFrame, otRadioFrame *aAckFrame, otError aError)
Pointer on receipt of a IEEE 802.15.4 frame.

typedef void(* otLinkRawEnergyScanDone)(otInstance *aInstance, int8_t aEnergyScanMaxRssi)
Pointer on receipt of a IEEE 802.15.4 frame.

Functions

otError otLinkRawSetReceiveDone(otInstance *aInstance, otLinkRawReceiveDone aCallback)
Enables/disables the raw link-layer.

bool otLinkRawIsEnabled(otInstance *aInstance)
Indicates whether or not the raw link-layer is enabled.

bool otLinkRawGetPromiscuous(otInstance *aInstance)
Gets the status of promiscuous mode.

otError otLinkRawSetPromiscuous(otInstance *aInstance, bool aEnable)
Enables or disables promiscuous mode.

otError otLinkRawSetShortAddress(otInstance *aInstance, uint16_t aShortAddress)
Set the Short Address for address filtering.

otError otLinkRawSleep(otInstance *aInstance)
Transition the radio from Receive to Sleep.

otError otLinkRawReceive(otInstance *aInstance)
Transitioning the radio from Sleep to Receive.

bool otLinkRawIsTransmittingOrScanning(otInstance *aInstance)
This function indicates whether or not the raw link-layer is busy transmitting or scanning.

otRadioFrame * otLinkRawGetTransmitBuffer(otInstance *aInstance)
The radio transitions from Transmit to Receive.

otError otLinkRawTransmit(otInstance *aInstance, otLinkRawTransmitDone aCallback)
Begins the transmit sequence on the radio.

int8_t otLinkRawGetRssi(otInstance *aInstance)
Get the most recent RSSI measurement.

otRadioCaps	otLinkRawGetCaps(otInstance *aInstance) Get the radio capabilities.
otError	otLinkRawEnergyScan(otInstance *aInstance, uint8_t aScanChannel, uint16_t aScanDuration, otLinkRawEnergyScanDone aCallback) Begins the energy scan sequence on the radio.
otError	otLinkRawSrcMatchEnable(otInstance *aInstance, bool aEnable) Enable/Disable source match for frame pending.
otError	otLinkRawSrcMatchAddShortEntry(otInstance *aInstance, uint16_t aShortAddress) Adding short address to the source match table.
otError	otLinkRawSrcMatchAddExtEntry(otInstance *aInstance, const otExtAddress *aExtAddress) Adding extended address to the source match table.
otError	otLinkRawSrcMatchClearShortEntry(otInstance *aInstance, uint16_t aShortAddress) Removing short address to the source match table.
otError	otLinkRawSrcMatchClearExtEntry(otInstance *aInstance, const otExtAddress *aExtAddress) Removing extended address to the source match table of the radio.
otError	otLinkRawSrcMatchClearShortEntries(otInstance *aInstance) Removing all the short addresses from the source match table.
otError	otLinkRawSrcMatchClearExtEntries(otInstance *aInstance) Removing all the extended addresses from the source match table.
otError	otLinkRawSetMacKey(otInstance *aInstance, uint8_t aKeyIdMode, uint8_t aKeyId, const otMacKey *aPrevKey, const otMacKey *aCurrKey, const otMacKey *aNextKey) Update MAC keys and key index.
otError	otLinkRawSetMacFrameCounter(otInstance *aInstance, uint32_t aMacFrameCounter) Sets the current MAC frame counter value.
otError	otLinkRawSetMacFrameCounterIfLarger(otInstance *aInstance, uint32_t aMacFrameCounter) Sets the current MAC frame counter value only if the new value is larger than the current one.
uint64_t	otLinkRawGetRadioTime(otInstance *aInstance) Get current platform time (64bits width) of the radio chip.

Typedef Documentation

otLinkRawReceiveDone

```
typedef void(* otLinkRawReceiveDone) (otInstance *aInstance, otRadioFrame *aFrame, otError aError) )(otInstance *aInstance, otRadioFrame *aFrame, otError aError)
```

Pointer on receipt of a IEEE 802.15.4 frame.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aFrame	A pointer to the received frame or NULL if the receive operation was aborted.
[in]	aError	OT_ERROR_NONE when successfully received a frame. OT_ERROR_ABORT when reception was aborted and a frame was not received.

Definition at line `63` of file `include/openthread/link_raw.h`

otLinkRawTransmitDone

```
typedef void(* otLinkRawTransmitDone) (otInstance *aInstance, otRadioFrame *aFrame, otRadioFrame *aAckFrame, otError aError) )(otInstance *aInstance, otRadioFrame *aFrame, otRadioFrame *aAckFrame, otError aError)
```

Pointer on receipt of a IEEE 802.15.4 frame.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aFrame	A pointer to the frame that was transmitted.
[in]	aAckFrame	A pointer to the ACK frame.
[in]	aError	OT_ERROR_NONE when the frame was transmitted. OT_ERROR_NO_ACK when the frame was transmitted but no ACK was received OT_ERROR_CHANNEL_ACCESS_FAILURE when the transmission could not take place due to activity on the channel. OT_ERROR_ABORT when transmission was aborted for other reasons.

Definition at line `188` of file `include/openthread/link_raw.h`

otLinkRawEnergyScanDone

```
typedef void(* otLinkRawEnergyScanDone) (otInstance *aInstance, int8_t aEnergyScanMaxRssi) )(otInstance *aInstance, int8_t aEnergyScanMaxRssi)
```

Pointer on receipt of a IEEE 802.15.4 frame.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnergyScanMaxRssi	The maximum RSSI encountered on the scanned channel.

Definition at line `239` of file `include/openthread/link_raw.h`

Function Documentation

otLinkRawSetReceiveDone

```
otError otLinkRawSetReceiveDone (otInstance *aInstance, otLinkRawReceiveDone aCallback)
```

Enables/disables the raw link-layer.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function called on receipt of a IEEE 802.15.4 frame. NULL to disable the raw-link layer.

Definition at line `77` of file `include/openthread/link_raw.h`

otLinkRawIsEnabled

```
bool otLinkRawIsEnabled (otInstance *aInstance)
```

Indicates whether or not the raw link-layer is enabled.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 88 of file include/openthread/link_raw.h

otLinkRawGetPromiscuous

```
bool otLinkRawGetPromiscuous (otInstance *aInstance)
```

Gets the status of promiscuous mode.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 99 of file include/openthread/link_raw.h

otLinkRawSetPromiscuous

```
otError otLinkRawSetPromiscuous (otInstance *aInstance, bool aEnable)
```

Enables or disables promiscuous mode.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnable	A value to enable or disable promiscuous mode.

Definition at line 111 of file include/openthread/link_raw.h

otLinkRawSetShortAddress

```
otError otLinkRawSetShortAddress (otInstance *aInstance, uint16_t aShortAddress)
```

Set the Short Address for address filtering.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aShortAddress	The IEEE 802.15.4 Short Address.

Definition at line 123 of file include/openthread/link_raw.h

otLinkRawSleep

```
otError otLinkRawSleep (otInstance *aInstance)
```

Transition the radio from Receive to Sleep.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Turn off the radio.

Definition at line 136 of file include/openthread/link_raw.h

otLinkRawReceive

```
otError otLinkRawReceive (otInstance *aInstance)
```

Transitioning the radio from Sleep to Receive.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Turn on the radio.

Definition at line 148 of file include/openthread/link_raw.h

otLinkRawIsTransmittingOrScanning

```
bool otLinkRawIsTransmittingOrScanning (otInstance *aInstance)
```

This function indicates whether or not the raw link-layer is busy transmitting or scanning.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 159 of file include/openthread/link_raw.h

otLinkRawGetTransmitBuffer

```
otRadioFrame * otLinkRawGetTransmitBuffer (otInstance *aInstance)
```

The radio transitions from Transmit to Receive.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns a pointer to the transmit buffer.

The caller forms the IEEE 802.15.4 frame in this buffer then calls otLinkRawTransmit() to request transmission.

Returns

- A pointer to the transmit buffer or NULL if the raw link-layer isn't enabled.

Definition at line 173 of file include/openthread/link_raw.h

otLinkRawTransmit

```
otError otLinkRawTransmit (otInstance *aInstance, otLinkRawTransmitDone aCallback)
```

Begins the transmit sequence on the radio.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[in]	aCallback	A pointer to a function called on completion of the transmission.

The caller must form the IEEE 802.15.4 frame in the buffer provided by otLinkRawGetTransmitBuffer() before requesting transmission. The channel and transmit power are also included in the otRadioFrame structure.

The transmit sequence consists of:

1. Transitioning the radio to Transmit from Receive.
2. Transmits the PSDU on the given channel and at the given transmit power.

Definition at line `210` of file `include/openthread/link_raw.h`

otLinkRawGetRssi

```
int8_t otLinkRawGetRssi (otInstance *aInstance)
```

Get the most recent RSSI measurement.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The RSSI in dBm when it is valid. 127 when RSSI is invalid.

Definition at line `220` of file `include/openthread/link_raw.h`

otLinkRawGetCaps

```
otRadioCaps otLinkRawGetCaps (otInstance *aInstance)
```

Get the radio capabilities.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The radio capability bit vector. The stack enables or disables some functions based on this value.

Definition at line `230` of file `include/openthread/link_raw.h`

otLinkRawEnergyScan

```
otError otLinkRawEnergyScan (otInstance *aInstance, uint8_t aScanChannel, uint16_t aScanDuration,
otLinkRawEnergyScanDone aCallback)
```

Begins the energy scan sequence on the radio.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aScanChannel	The channel to perform the energy scan on.
[in]	aScanDuration	The duration, in milliseconds, for the channel to be scanned.
[in]	aCallback	A pointer to a function called on completion of a scanned channel.

Definition at line 255 of file include/openthread/link_raw.h

otLinkRawSrcMatchEnable

> otError otLinkRawSrcMatchEnable (otInstance *aInstance, bool aEnable)

Enable/Disable source match for frame pending.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnable	Enable/disable source match for frame pending.

Definition at line 270 of file include/openthread/link_raw.h

otLinkRawSrcMatchAddShortEntry

> otError otLinkRawSrcMatchAddShortEntry (otInstance *aInstance, uint16_t aShortAddress)

Adding short address to the source match table.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aShortAddress	The short address to be added.

Definition at line 283 of file include/openthread/link_raw.h

otLinkRawSrcMatchAddExtEntry

> otError otLinkRawSrcMatchAddExtEntry (otInstance *aInstance, const otExtAddress *aExtAddress)

Adding extended address to the source match table.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	The extended address to be added.

Definition at line 296 of file include/openthread/link_raw.h

otLinkRawSrcMatchClearShortEntry

> otError otLinkRawSrcMatchClearShortEntry (otInstance *aInstance, uint16_t aShortAddress)

Removing short address to the source match table.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aShortAddress	The short address to be removed.

Definition at line 309 of file include/openthread/link_raw.h

otLinkRawSrcMatchClearExtEntry

```
otError otLinkRawSrcMatchClearExtEntry (otInstance *aInstance, const otExtAddress *aExtAddress)
```

Removing extended address to the source match table of the radio.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	The extended address to be removed.

Definition at line 322 of file include/openthread/link_raw.h

otLinkRawSrcMatchClearShortEntries

```
otError otLinkRawSrcMatchClearShortEntries (otInstance *aInstance)
```

Removing all the short addresses from the source match table.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 333 of file include/openthread/link_raw.h

otLinkRawSrcMatchClearExtEntries

```
otError otLinkRawSrcMatchClearExtEntries (otInstance *aInstance)
```

Removing all the extended addresses from the source match table.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 344 of file include/openthread/link_raw.h

otLinkRawSetMacKey

```
otError otLinkRawSetMacKey (otInstance *aInstance, uint8_t aKeyIdMode, uint8_t aKeyId, const otMacKey *aPrevKey,
const otMacKey *aCurrKey, const otMacKey *aNextKey)
```

Update MAC keys and key index.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aKeyIdMode	The key ID mode.
[in]	aKeyId	The key index.
[in]	aPrevKey	The previous MAC key.
[in]	aCurrKey	The current MAC key.
[in]	aNextKey	The next MAC key.

Definition at line `360` of file `include/openthread/link_raw.h`

otLinkRawSetMacFrameCounter

> otError otLinkRawSetMacFrameCounter (otInstance *aInstance, uint32_t aMacFrameCounter)

Sets the current MAC frame counter value.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMacFrameCounter	The MAC frame counter value.

Always sets the MAC counter to the new given value `aMacFrameCounter` independent of the current value.

Definition at line `380` of file `include/openthread/link_raw.h`

otLinkRawSetMacFrameCounterIfLarger

> otError otLinkRawSetMacFrameCounterIfLarger (otInstance *aInstance, uint32_t aMacFrameCounter)

Sets the current MAC frame counter value only if the new value is larger than the current one.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMacFrameCounter	The MAC frame counter value.

Definition at line `392` of file `include/openthread/link_raw.h`

otLinkRawGetRadioTime

> uint64_t otLinkRawGetRadioTime (otInstance *aInstance)

Get current platform time (64bits width) of the radio chip.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The current radio time in microseconds.

Definition at line `402` of file `include/openthread/link_raw.h`

Message

Message

This module includes functions that manipulate OpenThread message buffers.

Modules

otMessageSettings

otMessageQueue

otMessageQueueInfo

otBufferInfo

Enumerations

enum otMessagePriority {

OT_MESSAGE_PRIORITY_LOW = 0
OT_MESSAGE_PRIORITY_NORMAL = 1
OT_MESSAGE_PRIORITY_HIGH = 2

}

Defines the OpenThread message priority levels.

enum otMessageOrigin {

OT_MESSAGE_ORIGIN_THREAD_NETIF = 0
OT_MESSAGE_ORIGIN_HOST_TRUSTED = 1
OT_MESSAGE_ORIGIN_HOST_UNTRUSTED = 2

}

Defines the OpenThread message origins.

Typedefs

typedef struct otMessage — otMessage
An opaque representation of an OpenThread message buffer.

typedef enum otMessagePriority — otMessagePriority
Defines the OpenThread message priority levels.

typedef enum otMessageOrigin — otMessageOrigin
Defines the OpenThread message origins.

typedef struct otMessageSettings — otMessageSettings
Represents a message settings.

typedef struct otMessageQueueInfo — otMessageQueueInfo
Represents information about a message queue.

typedef struct otBufferInfo — otBufferInfo
Represents the message buffer information for different queues used by OpenThread stack.

Functions

void	otMessageFree(otMessage *aMessage) Free an allocated message buffer.
uint16_t	otMessageGetLength(const otMessage *aMessage) Get the message length in bytes.
otError	otMessageSetLength(otMessage *aMessage, uint16_t aLength) Set the message length in bytes.
uint16_t	otMessageGetOffset(const otMessage *aMessage) Get the message offset in bytes.
void	otMessageSetOffset(otMessage *aMessage, uint16_t aOffset) Set the message offset in bytes.
bool	otMessageIsLinkSecurityEnabled(const otMessage *aMessage) Indicates whether or not link security is enabled for the message.
bool	otMessageIsLoopbackToHostAllowed(const otMessage *aMessage) Indicates whether or not the message is allowed to be looped back to host.
void	otMessageSetLoopbackToHostAllowed(otMessage *aMessage, bool aAllowLoopbackToHost) Sets whether or not the message is allowed to be looped back to host.
otMessageOrigin	otMessageGetOrigin(const otMessage *aMessage) Gets the message origin.
void	otMessageSetOrigin(otMessage *aMessage, otMessageOrigin aOrigin) Sets the message origin.
void	otMessageSetDirectTransmission(otMessage *aMessage, bool aEnabled) Sets/forces the message to be forwarded using direct transmission.
int8_t	otMessageGetRss(const otMessage *aMessage) Returns the average RSS (received signal strength) associated with the message.
otError	otMessageAppend(otMessage *aMessage, const void *aBuf, uint16_t aLength) Append bytes to a message.
uint16_t	otMessageRead(const otMessage *aMessage, uint16_t aOffset, void *aBuf, uint16_t aLength) Read bytes from a message.
int	otMessageWrite(otMessage *aMessage, uint16_t aOffset, const void *aBuf, uint16_t aLength) Write bytes to a message.
void	otMessageQueueInit(otMessageQueue *aQueue) Initialize the message queue.
void	otMessageQueueEnqueue(otMessageQueue *aQueue, otMessage *aMessage) Adds a message to the end of the given message queue.
void	otMessageQueueEnqueueAtHead(otMessageQueue *aQueue, otMessage *aMessage) Adds a message at the head/front of the given message queue.
void	otMessageQueueDequeue(otMessageQueue *aQueue, otMessage *aMessage) Removes a message from the given message queue.
otMessage *	otMessageQueueGetHead(otMessageQueue *aQueue) Returns a pointer to the message at the head of the queue.

otMessage * otMessageQueueGetNext(otMessageQueue *aQueue, const otMessage *aMessage)
Returns a pointer to the next message in the queue by iterating forward (from head to tail).

void otMessageGetBufferInfo(otInstance *aInstance, otBufferInfo *aBufferInfo)
Get the Message Buffer information.

void otMessageResetBufferInfo(otInstance *aInstance)
Reset the Message Buffer information counter tracking the maximum number buffers in use at the same time.

Enumeration Documentation

otMessagePriority

> otMessagePriority

Defines the OpenThread message priority levels.

Enumerator	
OT_MESSAGE_PRIORITY_LOW	Low priority level.
OT_MESSAGE_PRIORITY_NORMAL	Normal priority level.
OT_MESSAGE_PRIORITY_HIGH	High priority level.

Definition at line `65` of file `include/openthread/message.h`

otMessageOrigin

> otMessageOrigin

Defines the OpenThread message origins.

Enumerator	
OT_MESSAGE_ORIGIN_THREAD_NETIF	Message from Thread Netif.
OT_MESSAGE_ORIGIN_HOST_TRUSTED	Message from a trusted source on host.
OT_MESSAGE_ORIGIN_HOST_UNTRUSTED	Message from an untrusted source on host.

Definition at line `76` of file `include/openthread/message.h`

Typedef Documentation

otMessage

> typedef struct otMessage otMessage

An opaque representation of an OpenThread message buffer.

Definition at line `59` of file `include/openthread/message.h`

otMessagePriority

> typedef enum otMessagePriority otMessagePriority

Defines the OpenThread message priority levels.

Definition at line 70 of file include/openthread/message.h

otMessageOrigin

```
typedef enum otMessageOrigin otMessageOrigin
```

Defines the OpenThread message origins.

Definition at line 81 of file include/openthread/message.h

otMessageSettings

```
typedef struct otMessageSettings otMessageSettings
```

Represents a message settings.

Definition at line 91 of file include/openthread/message.h

otMessageQueueInfo

```
typedef struct otMessageQueueInfo otMessageQueueInfo
```

Represents information about a message queue.

Definition at line 332 of file include/openthread/message.h

otBufferInfo

```
typedef struct otBufferInfo otBufferInfo
```

Represents the message buffer information for different queues used by OpenThread stack.

Definition at line 358 of file include/openthread/message.h

Function Documentation

otMessageFree

```
void otMessageFree (otMessage *aMessage)
```

Free an allocated message buffer.

Parameters

[in]	aMessage	A pointer to a message buffer.

See Also

- otMessageAppend
- otMessageGetLength
- otMessageSetLength
- otMessageGetOffset
- otMessageSetOffset

otMessageRead
- otMessageWrite

Definition at line 107 of file include/openthread/message.h

otMessageGetLength

```
uint16_t otMessageGetLength (const otMessage *aMessage)
```

Get the message length in bytes.

Parameters

[in]	aMessage	A pointer to a message buffer.

Returns

- The message length in bytes.

See Also

- otMessageFree
- otMessageAppend
- otMessageSetLength
- otMessageGetOffset
- otMessageSetOffset
- otMessageRead
- otMessageWrite
- otMessageSetLength

Definition at line 126 of file include/openthread/message.h

otMessageSetLength

```
otError otMessageSetLength (otMessage *aMessage, uint16_t aLength)
```

Set the message length in bytes.

Parameters

[in]	aMessage	A pointer to a message buffer.
[in]	aLength	A length in bytes.

See Also

- otMessageFree
- otMessageAppend
- otMessageGetLength
- otMessageGetOffset
- otMessageSetOffset
- otMessageRead
- otMessageWrite

Definition at line 146 of file include/openthread/message.h

otMessageGetOffset

```
uint16_t otMessageGetOffset (const otMessage *aMessage)
```

Get the message offset in bytes.

Parameters

[in]	aMessage	A pointer to a message buffer.

Returns

- The message offset value.

See Also

- otMessageFree
- otMessageAppend
- otMessageGetLength
- otMessageSetLength
- otMessageSetOffset
- otMessageRead
- otMessageWrite

Definition at line 164 of file include/openthread/message.h

otMessageSetOffset

```
void otMessageSetOffset (otMessage *aMessage, uint16_t aOffset)
```

Set the message offset in bytes.

Parameters

[in]	aMessage	A pointer to a message buffer.
[in]	aOffset	An offset in bytes.

See Also

- otMessageFree
- otMessageAppend
- otMessageGetLength
- otMessageSetLength
- otMessageGetOffset
- otMessageRead
- otMessageWrite

Definition at line 181 of file include/openthread/message.h

otMessageIsLinkSecurityEnabled

```
bool otMessageIsLinkSecurityEnabled (const otMessage *aMessage)
```

Indicates whether or not link security is enabled for the message.

Parameters

[in]	aMessage	A pointer to a message buffer.

Definition at line 192 of file include/openthread/message.h

otMessageIsLoopbackToHostAllowed

> bool otMessageIsLoopbackToHostAllowed (const otMessage *aMessage)

Indicates whether or not the message is allowed to be looped back to host.

Parameters

[in]	aMessage	A pointer to a message buffer.

Definition at line `203` of file `include/openthread/message.h`

otMessageSetLoopbackToHostAllowed

> void otMessageSetLoopbackToHostAllowed (otMessage *aMessage, bool aAllowLoopbackToHost)

Sets whether or not the message is allowed to be looped back to host.

Parameters

[in]	aMessage	A pointer to a message buffer.
[in]	aAllowLoopbackToHost	Whether to allow the message to be looped back to host.

Definition at line `212` of file `include/openthread/message.h`

otMessageGetOrigin

> otMessageOrigin otMessageGetOrigin (const otMessage *aMessage)

Gets the message origin.

Parameters

[in]	aMessage	A pointer to a message buffer.

Returns

- The message origin.

Definition at line `222` of file `include/openthread/message.h`

otMessageSetOrigin

> void otMessageSetOrigin (otMessage *aMessage, otMessageOrigin aOrigin)

Sets the message origin.

Parameters

[in]	aMessage	A pointer to a message buffer.
[in]	aOrigin	The message origin.

Definition at line `231` of file `include/openthread/message.h`

otMessageSetDirectTransmission

> void otMessageSetDirectTransmission (otMessage *aMessage, bool aEnabled)

Sets/forces the message to be forwarded using direct transmission.

Parameters

[in]	aMessage	A pointer to a message buffer.
[in]	aEnabled	If `true`, the message is forced to use direct transmission. If `false`, the message follows the normal procedure.

Default setting for a new message is `false`.

Definition at line `242` of file `include/openthread/message.h`

otMessageGetRss

> int8_t otMessageGetRss (const otMessage *aMessage)

Returns the average RSS (received signal strength) associated with the message.

Parameters

N/A	aMessage	

Returns

- The average RSS value (in dBm) or OT_RADIO_RSSI_INVALID if no average RSS is available.

Definition at line `250` of file `include/openthread/message.h`

otMessageAppend

> otError otMessageAppend (otMessage *aMessage, const void *aBuf, uint16_t aLength)

Append bytes to a message.

Parameters

[in]	aMessage	A pointer to a message buffer.
[in]	aBuf	A pointer to the data to append.
[in]	aLength	Number of bytes to append.

See Also

- otMessageFree
- otMessageGetLength
- otMessageSetLength
- otMessageGetOffset
- otMessageSetOffset
- otMessageRead
- otMessageWrite

Definition at line `271` of file `include/openthread/message.h`

otMessageRead

> uint16_t otMessageRead (const otMessage *aMessage, uint16_t aOffset, void *aBuf, uint16_t aLength)

Read bytes from a message.

Parameters

[in]	aMessage	A pointer to a message buffer.
[in]	aOffset	An offset in bytes.
[in]	aBuf	A pointer to a buffer that message bytes are read to.
[in]	aLength	Number of bytes to read.

Returns

- The number of bytes read.

See Also

- otMessageFree
- otMessageAppend
- otMessageGetLength
- otMessageSetLength
- otMessageGetOffset
- otMessageSetOffset
- otMessageWrite

Definition at line `292` of file `include/openthread/message.h`

otMessageWrite

> int otMessageWrite (otMessage *aMessage, uint16_t aOffset, const void *aBuf, uint16_t aLength)

Write bytes to a message.

Parameters

[in]	aMessage	A pointer to a message buffer.
[in]	aOffset	An offset in bytes.
[in]	aBuf	A pointer to a buffer that message bytes are written from.
[in]	aLength	Number of bytes to write.

Returns

- The number of bytes written.

See Also

- otMessageFree
- otMessageAppend
- otMessageGetLength
- otMessageSetLength
- otMessageGetOffset
- otMessageSetOffset
- otMessageRead

Definition at line `313` of file `include/openthread/message.h`

otMessageQueueInit

> void otMessageQueueInit (otMessageQueue *aQueue)

Initialize the message queue.

Parameters

[in]	aQueue	A pointer to a message queue.

MUST be called once and only once for a `otMessageQueue` instance before any other `otMessageQueue` functions. The behavior is undefined if other queue APIs are used with an `otMessageQueue` before it being initialized or if it is initialized more than once.

Definition at line `370` of file `include/openthread/message.h`

otMessageQueueEnqueue

> void otMessageQueueEnqueue (otMessageQueue *aQueue, otMessage *aMessage)

Adds a message to the end of the given message queue.

Parameters

[in]	aQueue	A pointer to the message queue.
[in]	aMessage	The message to add.

Definition at line `379` of file `include/openthread/message.h`

otMessageQueueEnqueueAtHead

> void otMessageQueueEnqueueAtHead (otMessageQueue *aQueue, otMessage *aMessage)

Adds a message at the head/front of the given message queue.

Parameters

[in]	aQueue	A pointer to the message queue.
[in]	aMessage	The message to add.

Definition at line `388` of file `include/openthread/message.h`

otMessageQueueDequeue

> void otMessageQueueDequeue (otMessageQueue *aQueue, otMessage *aMessage)

Removes a message from the given message queue.

Parameters

[in]	aQueue	A pointer to the message queue.
[in]	aMessage	The message to remove.

Definition at line `397` of file `include/openthread/message.h`

otMessageQueueGetHead

```
otMessage * otMessageQueueGetHead (otMessageQueue *aQueue)
```

Returns a pointer to the message at the head of the queue.

Parameters

[in]	aQueue	A pointer to a message queue.

Returns

- A pointer to the message at the head of queue or NULL if queue is empty.

Definition at line `407` of file `include/openthread/message.h`

otMessageQueueGetNext

```
otMessage * otMessageQueueGetNext (otMessageQueue *aQueue, const otMessage *aMessage)
```

Returns a pointer to the next message in the queue by iterating forward (from head to tail).

Parameters

[in]	aQueue	A pointer to a message queue.
[in]	aMessage	A pointer to current message buffer.

Returns

- A pointer to the next message in the queue after `aMessage` or NULL if `aMessage is the tail of queue. NULL is returned if` aMessage `is not in the queue` aQueue`.

Definition at line `419` of file `include/openthread/message.h`

otMessageGetBufferInfo

```
void otMessageGetBufferInfo (otInstance *aInstance, otBufferInfo *aBufferInfo)
```

Get the Message Buffer information.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[out]	aBufferInfo	A pointer where the message buffer information is written.

Definition at line `428` of file `include/openthread/message.h`

otMessageResetBufferInfo

```
void otMessageResetBufferInfo (otInstance *aInstance)
```

Reset the Message Buffer information counter tracking the maximum number buffers in use at the same time.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

This resets `mMaxUsedBuffers` in `otBufferInfo`.

Definition at line `438` of file `include/openthread/message.h`

otMessageSettings

Represents a message settings.

Public Attributes

bool	mLinkSecurityEnabled TRUE if the message should be secured at Layer 2.
uint8_t	mPriority Priority level (MUST be a `OT_MESSAGE_PRIORITY_*` from `otMessagePriority`).

Public Attribute Documentation

mLinkSecurityEnabled

```
bool otMessageSettings::mLinkSecurityEnabled
```

TRUE if the message should be secured at Layer 2.

Definition at line `89` of file `include/openthread/message.h`

mPriority

```
uint8_t otMessageSettings::mPriority
```

Priority level (MUST be a `OT_MESSAGE_PRIORITY_*` from `otMessagePriority`).

Definition at line `90` of file `include/openthread/message.h`

otMessageQueue

Represents an OpenThread message queue.

Public Attributes

void *	mData Opaque data used by the implementation.

Public Attribute Documentation

mData

```
void* otMessageQueue::mData
```

Opaque data used by the implementation.

Definition at line 320 of file include/openthread/message.h

otMessageQueueInfo

Represents information about a message queue.

Public Attributes

uint16_t	mNumMessages Number of messages in the queue.
uint16_t	mNumBuffers Number of data buffers used by messages in the queue.
uint32_t	mTotalBytes Total number of bytes used by all messages in the queue.

Public Attribute Documentation

mNumMessages

```
uint16_t otMessageQueueInfo::mNumMessages
```

Number of messages in the queue.

Definition at line `329` of file `include/openthread/message.h`

mNumBuffers

```
uint16_t otMessageQueueInfo::mNumBuffers
```

Number of data buffers used by messages in the queue.

Definition at line `330` of file `include/openthread/message.h`

mTotalBytes

```
uint32_t otMessageQueueInfo::mTotalBytes
```

Total number of bytes used by all messages in the queue.

Definition at line `331` of file `include/openthread/message.h`

otBufferInfo

Represents the message buffer information for different queues used by OpenThread stack.

Public Attributes

uint16_t	mTotalBuffers The total number of buffers in the messages pool (0xffff if unknown).
uint16_t	mFreeBuffers The number of free buffers (0xffff if unknown).
uint16_t	mMaxUsedBuffers The maximum number of used buffers at the same time since OT stack initialization or last call to otMessageResetBufferInfo().
otMessageQueueInfo	m6loSendQueue Info about 6LoWPAN send queue.
otMessageQueueInfo	m6loReassemblyQueue Info about 6LoWPAN reassembly queue.
otMessageQueueInfo	mIp6Queue Info about IPv6 send queue.
otMessageQueueInfo	mMplQueue Info about MPL send queue.
otMessageQueueInfo	mMleQueue Info about MLE delayed message queue.
otMessageQueueInfo	mCoapQueue Info about CoAP/TMF send queue.
otMessageQueueInfo	mCoapSecureQueue Info about CoAP secure send queue.
otMessageQueueInfo	mApplicationCoapQueue Info about application CoAP send queue.

Public Attribute Documentation

mTotalBuffers

```
uint16_t otBufferInfo::mTotalBuffers
```

The total number of buffers in the messages pool (0xffff if unknown).

Definition at line 340 of file include/openthread/message.h

mFreeBuffers

```
uint16_t otBufferInfo::mFreeBuffers
```

The number of free buffers (0xffff if unknown).

Definition at line 341 of file include/openthread/message.h

mMaxUsedBuffers

```
uint16_t otBufferInfo::mMaxUsedBuffers
```

The maximum number of used buffers at the same time since OT stack initialization or last call to otMessageResetBufferInfo() .

Definition at line 348 of file include/openthread/message.h

m6loSendQueue

```
otMessageQueueInfo otBufferInfo::m6loSendQueue
```

Info about 6LoWPAN send queue.

Definition at line 350 of file include/openthread/message.h

m6loReassemblyQueue

```
otMessageQueueInfo otBufferInfo::m6loReassemblyQueue
```

Info about 6LoWPAN reassembly queue.

Definition at line 351 of file include/openthread/message.h

mIp6Queue

```
otMessageQueueInfo otBufferInfo::mIp6Queue
```

Info about IPv6 send queue.

Definition at line 352 of file include/openthread/message.h

mMplQueue

```
otMessageQueueInfo otBufferInfo::mMplQueue
```

Info about MPL send queue.

Definition at line 353 of file include/openthread/message.h

mMleQueue

```
otMessageQueueInfo otBufferInfo::mMleQueue
```

Info about MLE delayed message queue.

Definition at line 354 of file include/openthread/message.h

mCoapQueue

> otMessageQueueInfo otBufferInfo::mCoapQueue

Info about CoAP/TMF send queue.

Definition at line 355 of file include/openthread/message.h

mCoapSecureQueue

> otMessageQueueInfo otBufferInfo::mCoapSecureQueue

Info about CoAP secure send queue.

Definition at line 356 of file include/openthread/message.h

mApplicationCoapQueue

> otMessageQueueInfo otBufferInfo::mApplicationCoapQueue

Info about application CoAP send queue.

Definition at line 357 of file include/openthread/message.h

Multi Radio Link

Multi Radio Link

This module includes definitions and functions for multi radio link.

Modules

otRadioLinkInfo

otMultiRadioNeighborInfo

Typedefs

typedef struct otRadioLinkInfo	otRadioLinkInfo Represents information associated with a radio link.
typedef struct otMultiRadioNeighborInfo	otMultiRadioNeighborInfo Represents multi radio link information associated with a neighbor.

Functions

otError	otMultiRadioGetNeighborInfo(otInstance *aInstance, const otExtAddress *aExtAddress, otMultiRadioNeighborInfo *aNeighborInfo) Gets the multi radio link information associated with a neighbor with a given Extended Address.

Typedef Documentation

otRadioLinkInfo

```
typedef struct otRadioLinkInfo otRadioLinkInfo
```

Represents information associated with a radio link.

Definition at line 61 of file include/openthread/multi_radio.h

otMultiRadioNeighborInfo

```
typedef struct otMultiRadioNeighborInfo otMultiRadioNeighborInfo
```

Represents multi radio link information associated with a neighbor.

Definition at line 73 of file include/openthread/multi_radio.h

Function Documentation

otMultiRadioGetNeighborInfo

> otError otMultiRadioGetNeighborInfo (otInstance *aInstance, const otExtAddress *aExtAddress, otMultiRadioNeighborInfo *aNeighborInfo)

Gets the multi radio link information associated with a neighbor with a given Extended Address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	The Extended Address of neighbor.
[out]	aNeighborInfo	A pointer to `otMultiRadioNeighborInfo` to output the neighbor info (on success).

`OPENTHREAD_CONFIG_MULTI_RADIO` must be enabled.

Definition at line `88` of file `include/openthread/multi_radio.h`

otRadioLinkInfo

Represents information associated with a radio link.

Public Attributes

uint8_t	mPreference Preference level of radio link.

Public Attribute Documentation

mPreference

```
uint8_t otRadioLinkInfo::mPreference
```

Preference level of radio link.

Definition at line 60 of file include/openthread/multi_radio.h

otMultiRadioNeighborInfo

Represents multi radio link information associated with a neighbor.

Public Attributes

bool	mSupportsIeee802154 Neighbor supports IEEE 802.15.4 radio link.
bool	mSupportsTrelUdp6 Neighbor supports Thread Radio Encapsulation Link (TREL) radio link.
otRadioLinkInfo	mIeee802154Info Additional info for 15.4 radio link (applicable when supported).
otRadioLinkInfo	mTrelUdp6Info Additional info for TREL radio link (applicable when supported).

Public Attribute Documentation

mSupportsIeee802154

```
bool otMultiRadioNeighborInfo::mSupportsIeee802154
```

Neighbor supports IEEE 802.15.4 radio link.

Definition at line `69` of file `include/openthread/multi_radio.h`

mSupportsTrelUdp6

```
bool otMultiRadioNeighborInfo::mSupportsTrelUdp6
```

Neighbor supports Thread Radio Encapsulation Link (TREL) radio link.

Definition at line `70` of file `include/openthread/multi_radio.h`

mIeee802154Info

```
otRadioLinkInfo otMultiRadioNeighborInfo::mIeee802154Info
```

Additional info for 15.4 radio link (applicable when supported).

Definition at line `71` of file `include/openthread/multi_radio.h`

mTrelUdp6Info

```
otRadioLinkInfo otMultiRadioNeighborInfo::mTrelUdp6Info
```

Additional info for TREL radio link (applicable when supported).

Definition at line 72 of file include/openthread/multi_radio.h

TREL - Thread Stack

This module defines Thread Radio Encapsulation Link (TREL) APIs for Thread Over Infrastructure.

The functions in this module require `OPENTHREAD_CONFIG_RADIO_LINK_TREL_ENABLE` to be enabled.

Modules

otTrelPeer

Typedefs

typedef struct otTrelPeer	otTrelPeer Represents a TREL peer.
typedef uint16_t	otTrelPeerIterator Represents an iterator for iterating over TREL peer table entries.

Functions

void	otTrelSetEnabled(otInstance *aInstance, bool aEnable) Enables or disables TREL operation.
bool	otTrelIsEnabled(otInstance *aInstance) Indicates whether the TREL operation is enabled.
void	otTrelInitPeerIterator(otInstance *aInstance, otTrelPeerIterator *aIterator) Initializes a peer table iterator.
const otTrelPeer *	otTrelGetNextPeer(otInstance *aInstance, otTrelPeerIterator *aIterator) Iterates over the peer table entries and get the next entry from the table.
void	otTrelSetFilterEnabled(otInstance *aInstance, bool aEnable) Sets the filter mode (enables/disables filtering).
bool	otTrelIsFilterEnabled(otInstance *aInstance) Indicates whether or not the filter mode is enabled.

Typedef Documentation

otTrelPeer

```
typedef struct otTrelPeer otTrelPeer
```

Represents a TREL peer.

Definition at line 68 of file include/openthread/trel.h

otTrelPeerIterator

```
typedef uint16_t otTrelPeerIterator
```

Represents an iterator for iterating over TREL peer table entries.

Definition at line `74` of file `include/openthread/trel.h`

Function Documentation

otTrelSetEnabled

```
void otTrelSetEnabled (otInstance *aInstance, bool aEnable)
```

Enables or disables TREL operation.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnable	A boolean to enable/disable the TREL operation.

When `aEnable` is true, this function initiates an ongoing DNS-SD browse on the service name "_trel._udp" within the local browsing domain to discover other devices supporting TREL. Device also registers a new service to be advertised using DNS-SD, with the service name is "_trel._udp" indicating its support for TREL. Device is then ready to receive TREL messages from peers.

When `aEnable` is false, this function stops the DNS-SD browse on the service name "_trel._udp", stops advertising TREL DNS-SD service, and clears the TREL peer table.

Note

- By default the OpenThread stack enables the TREL operation on start.

Definition at line `93` of file `include/openthread/trel.h`

otTrelIsEnabled

```
bool otTrelIsEnabled (otInstance *aInstance)
```

Indicates whether the TREL operation is enabled.

Parameters

[in]	aInstance	The OpenThread instance.

Definition at line `104` of file `include/openthread/trel.h`

otTrelInitPeerIterator

```
void otTrelInitPeerIterator (otInstance *aInstance, otTrelPeerIterator *aIterator)
```

Initializes a peer table iterator.

Parameters

[in]	aInstance	The OpenThread instance.
[in]	aIterator	The iterator to initialize.

Definition at line 113 of file include/openthread/trel.h

otTrelGetNextPeer

```
const otTrelPeer * otTrelGetNextPeer (otInstance *aInstance, otTrelPeerIterator *aIterator)
```

Iterates over the peer table entries and get the next entry from the table.

Parameters

[in]	aInstance	The OpenThread instance.
[in]	aIterator	The iterator. MUST be initialized.

Returns

- A pointer to the next otTrelPeer entry or NULL if no more entries in the table.

Definition at line 124 of file include/openthread/trel.h

otTrelSetFilterEnabled

```
void otTrelSetFilterEnabled (otInstance *aInstance, bool aEnable)
```

Sets the filter mode (enables/disables filtering).

Parameters

[in]	aInstance	The OpenThread instance.
[in]	aEnable	TRUE to enable filter mode, FALSE to disable filter mode.

When filter mode is enabled, any rx and tx traffic through TREL interface is silently dropped. This is mainly intended for use during testing.

Unlike otTrel{Enable/Disable}() which fully starts/stops the TREL operation, when filter mode is enabled the TREL interface continues to be enabled.

Definition at line 139 of file include/openthread/trel.h

otTrelIsFilterEnabled

```
bool otTrelIsFilterEnabled (otInstance *aInstance)
```

Indicates whether or not the filter mode is enabled.

Parameters

[in]	aInstance	The OpenThread instance.

Definition at line 150 of file include/openthread/trel.h

otTrelPeer

Represents a TREL peer.

Public Attributes

otExtAddress	mExtAddress The Extended MAC Address of TREL peer.
otExtendedPanId	mExtPanId The Extended PAN Identifier of TREL peer.
otSockAddr	mSockAddr The IPv6 socket address of TREL peer.

Public Attribute Documentation

mExtAddress

```
otExtAddress otTrelPeer::mExtAddress
```

The Extended MAC Address of TREL peer.

Definition at line `65` of file `include/openthread/trel.h`

mExtPanId

```
otExtendedPanId otTrelPeer::mExtPanId
```

The Extended PAN Identifier of TREL peer.

Definition at line `66` of file `include/openthread/trel.h`

mSockAddr

```
otSockAddr otTrelPeer::mSockAddr
```

The IPv6 socket address of TREL peer.

Definition at line `67` of file `include/openthread/trel.h`

Thread

Thread

Modules

- Backbone Router
- Border Agent
- Border Router
- Border Routing Manager
- Commissioner
- General
- Joiner
- Operational Dataset
- Router/Leader
- Server

Backbone Router

Backbone Router

This module includes functions for the OpenThread Backbone Router Service.

Modules

otBackboneRouterConfig

otBackboneRouterMulticastListenerInfo

otBackboneRouterNdProxyInfo

Enumerations

enum otBackboneRouterState {

OT_BACKBONE_ROUTER_STATE_DISABLED = 0
OT_BACKBONE_ROUTER_STATE_SECONDARY = 1
OT_BACKBONE_ROUTER_STATE_PRIMARY = 2

}

Represents the Backbone Router Status.

enum otBackboneRouterMulticastListenerEvent {

OT_BACKBONE_ROUTER_MULTICAST_LISTENER_ADDED = 0
OT_BACKBONE_ROUTER_MULTICAST_LISTENER_REMOVED = 1

}

Represents the Multicast Listener events.

enum otBackboneRouterNdProxyEvent {

OT_BACKBONE_ROUTER_NDPROXY_ADDED = 0
OT_BACKBONE_ROUTER_NDPROXY_REMOVED = 1
OT_BACKBONE_ROUTER_NDPROXY_RENEWED = 2
OT_BACKBONE_ROUTER_NDPROXY_CLEARED = 3

}

Represents the ND Proxy events.

enum otBackboneRouterDomainPrefixEvent {

OT_BACKBONE_ROUTER_DOMAIN_PREFIX_ADDED = 0
OT_BACKBONE_ROUTER_DOMAIN_PREFIX_REMOVED = 1
OT_BACKBONE_ROUTER_DOMAIN_PREFIX_CHANGED = 2

}

Represents the Domain Prefix events.

Typedefs

typedef struct otBackboneRouterConfig otBackboneRouterConfig

Represents Backbone Router configuration.

typedef void(* otBackboneRouterMulticastListenerCallback)(void *aContext, otBackboneRouterMulticastListenerEvent aEvent, const otIp6Address *aAddress)
Pointer is called whenever the Multicast Listeners change.

typedef uint16_t otBackboneRouterMulticastListenerIterator
Used to iterate through Multicast Listeners.

typedef struct otBackboneRouterMulticastListenerInfo otBackboneRouterMulticastListenerInfo
Represents a Backbone Router Multicast Listener info.

typedef void(* otBackboneRouterNdProxyCallback)(void *aContext, otBackboneRouterNdProxyEvent aEvent, const otIp6Address *aDua)
Pointer is called whenever the Nd Proxy changed.

typedef struct otBackboneRouterNdProxyInfo otBackboneRouterNdProxyInfo
Represents the Backbone Router ND Proxy info.

typedef void(* otBackboneRouterDomainPrefixCallback)(void *aContext, otBackboneRouterDomainPrefixEvent aEvent, const otIp6Prefix *aDomainPrefix)
Pointer is called whenever the Domain Prefix changed.

Functions

otError otBackboneRouterGetPrimary(otInstance *aInstance, otBackboneRouterConfig *aConfig)
Gets the Primary Backbone Router information in the Thread Network.

void otBackboneRouterSetEnabled(otInstance *aInstance, bool aEnable)
Enables or disables Backbone functionality.

otBackboneRouterState otBackboneRouterGetState(otInstance *aInstance)
Gets the Backbone Router otBackboneRouterState.

void otBackboneRouterGetConfig(otInstance *aInstance, otBackboneRouterConfig *aConfig)
Gets the local Backbone Router configuration.

otError otBackboneRouterSetConfig(otInstance *aInstance, const otBackboneRouterConfig *aConfig)
Sets the local Backbone Router configuration otBackboneRouterConfig.

otError otBackboneRouterRegister(otInstance *aInstance)
Explicitly registers local Backbone Router configuration.

uint8_t otBackboneRouterGetRegistrationJitter(otInstance *aInstance)
Returns the Backbone Router registration jitter value.

void otBackboneRouterSetRegistrationJitter(otInstance *aInstance, uint8_t aJitter)
Sets the Backbone Router registration jitter value.

otError otBackboneRouterGetDomainPrefix(otInstance *aInstance, otBorderRouterConfig *aConfig)
Gets the local Domain Prefix configuration.

void otBackboneRouterConfigNextDuaRegistrationResponse(otInstance *aInstance, const otIp6InterfaceIdentifier *aMlIid, uint8_t aStatus)
Configures response status for next DUA registration.

void otBackboneRouterConfigNextMulticastListenerRegistrationResponse(otInstance *aInstance, uint8_t aStatus)
Configures the response status for the next Multicast Listener Registration.

void otBackboneRouterSetMulticastListenerCallback(otInstance *aInstance, otBackboneRouterMulticastListenerCallback aCallback, void *aContext)
Sets the Backbone Router Multicast Listener callback.

void otBackboneRouterMulticastListenerClear(otInstance *aInstance)
Clears the Multicast Listeners.

otError otBackboneRouterMulticastListenerAdd(otInstance *aInstance, const otIp6Address *aAddress, uint32_t aTimeout)
Adds a Multicast Listener with a timeout value, in seconds.

otError otBackboneRouterMulticastListenerGetNext(otInstance *aInstance, otBackboneRouterMulticastListenerIterator *aIterator, otBackboneRouterMulticastListenerInfo *aListenerInfo)
Gets the next Multicast Listener info (using an iterator).

void otBackboneRouterSetNdProxyCallback(otInstance *aInstance, otBackboneRouterNdProxyCallback aCallback, void *aContext)
Sets the Backbone Router ND Proxy callback.

otError otBackboneRouterGetNdProxyInfo(otInstance *aInstance, const otIp6Address *aDua, otBackboneRouterNdProxyInfo *aNdProxyInfo)
Gets the Backbone Router ND Proxy info.

void otBackboneRouterSetDomainPrefixCallback(otInstance *aInstance, otBackboneRouterDomainPrefixCallback aCallback, void *aContext)
Sets the Backbone Router Domain Prefix callback.

Macros

#define OT_BACKBONE_ROUTER_MULTICAST_LISTENER_ITERATOR_INIT 0
Initializer for otBackboneRouterMulticastListenerIterator.

Enumeration Documentation

otBackboneRouterState

```
otBackboneRouterState
```

Represents the Backbone Router Status.

Enumerator	
OT_BACKBONE_ROUTER_STATE_DISABLED	Backbone function is disabled.
OT_BACKBONE_ROUTER_STATE_SECONDARY	Secondary Backbone Router.
OT_BACKBONE_ROUTER_STATE_PRIMARY	The Primary Backbone Router.

Definition at line 59 of file include/openthread/backbone_router_ftd.h

otBackboneRouterMulticastListenerEvent

```
otBackboneRouterMulticastListenerEvent
```

Represents the Multicast Listener events.

Enumerator	
OT_BACKBONE_ROUTER_MULTICAST_LISTENER_ADDED	Multicast Listener was added.
OT_BACKBONE_ROUTER_MULTICAST_LISTENER_REMOVED	Multicast Listener was removed or expired.

Definition at line 231 of file include/openthread/backbone_router_ftd.h

otBackboneRouterNdProxyEvent

> otBackboneRouterNdProxyEvent

Represents the ND Proxy events.

Enumerator	
OT_BACKBONE_ROUTER_NDPROXY_ADDED	ND Proxy was added.
OT_BACKBONE_ROUTER_NDPROXY_REMOVED	ND Proxy was removed.
OT_BACKBONE_ROUTER_NDPROXY_RENEWED	ND Proxy was renewed.
OT_BACKBONE_ROUTER_NDPROXY_CLEARED	All ND Proxies were cleared.

Definition at line 340 of file include/openthread/backbone_router_ftd.h

otBackboneRouterDomainPrefixEvent

> otBackboneRouterDomainPrefixEvent

Represents the Domain Prefix events.

Enumerator	
OT_BACKBONE_ROUTER_DOMAIN_PREFIX_ADDED	Domain Prefix was added.
OT_BACKBONE_ROUTER_DOMAIN_PREFIX_REMOVED	Domain Prefix was removed.
OT_BACKBONE_ROUTER_DOMAIN_PREFIX_CHANGED	Domain Prefix was changed.

Definition at line 403 of file include/openthread/backbone_router_ftd.h

Typedef Documentation

otBackboneRouterConfig

> typedef struct otBackboneRouterConfig otBackboneRouterConfig

Represents Backbone Router configuration.

Definition at line 64 of file include/openthread/backbone_router.h

otBackboneRouterMulticastListenerCallback

> typedef void(* otBackboneRouterMulticastListenerCallback) (void *aContext, otBackboneRouterMulticastListenerEvent aEvent, const otIp6Address *aAddress))(void *aContext, otBackboneRouterMulticastListenerEvent aEvent, const otIp6Address *aAddress)

Pointer is called whenever the Multicast Listeners change.

Parameters

[in]	aContext	The user context pointer.
[in]	aEvent	The Multicast Listener event.
[in]	aAddress	The IPv6 multicast address of the Multicast Listener.

Definition at line 245 of file include/openthread/backbone_router_ftd.h

otBackboneRouterMulticastListenerIterator

```
typedef uint16_t otBackboneRouterMulticastListenerIterator
```

Used to iterate through Multicast Listeners.

Definition at line 302 of file include/openthread/backbone_router_ftd.h

otBackboneRouterMulticastListenerInfo

```
typedef struct otBackboneRouterMulticastListenerInfo otBackboneRouterMulticastListenerInfo
```

Represents a Backbone Router Multicast Listener info.

Definition at line 312 of file include/openthread/backbone_router_ftd.h

otBackboneRouterNdProxyCallback

```
typedef void(* otBackboneRouterNdProxyCallback) (void *aContext, otBackboneRouterNdProxyEvent aEvent, const otIp6Address *aDua) )(void *aContext, otBackboneRouterNdProxyEvent aEvent, const otIp6Address *aDua)
```

Pointer is called whenever the Nd Proxy changed.

Parameters

[in]	aContext	The user context pointer.
[in]	aEvent	The ND Proxy event.
[in]	aDua	The Domain Unicast Address of the ND Proxy, or `nullptr` if `aEvent` is `OT_BACKBONE_ROUTER_NDPROXY_CLEARED`.

Definition at line 357 of file include/openthread/backbone_router_ftd.h

otBackboneRouterNdProxyInfo

```
typedef struct otBackboneRouterNdProxyInfo otBackboneRouterNdProxyInfo
```

Represents the Backbone Router ND Proxy info.

Definition at line 382 of file include/openthread/backbone_router_ftd.h

otBackboneRouterDomainPrefixCallback

```
typedef void(* otBackboneRouterDomainPrefixCallback) (void *aContext, otBackboneRouterDomainPrefixEvent aEvent, const otIp6Prefix *aDomainPrefix) )(void *aContext, otBackboneRouterDomainPrefixEvent aEvent, const otIp6Prefix *aDomainPrefix)
```

Pointer is called whenever the Domain Prefix changed.

Parameters

[in]	aContext	The user context pointer.
[in]	aEvent	The Domain Prefix event.
[in]	aDomainPrefix	The new Domain Prefix if added or changed, nullptr otherwise.

Definition at line `418` of file `include/openthread/backbone_router_ftd.h`

Function Documentation

otBackboneRouterGetPrimary

> otError otBackboneRouterGetPrimary (otInstance *aInstance, otBackboneRouterConfig *aConfig)

Gets the Primary Backbone Router information in the Thread Network.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aConfig	A pointer to where to put Primary Backbone Router information.

Definition at line `76` of file `include/openthread/backbone_router.h`

otBackboneRouterSetEnabled

> void otBackboneRouterSetEnabled (otInstance *aInstance, bool aEnable)

Enables or disables Backbone functionality.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnable	TRUE to enable Backbone functionality, FALSE otherwise.

If enabled, a Server Data Request message `SRV_DATA.ntf` is triggered for the attached device if there is no Backbone Router Service in the Thread Network Data.

If disabled, `SRV_DATA.ntf` is triggered if the Backbone Router is in the Primary state.

Available when `OPENTHREAD_CONFIG_BACKBONE_ROUTER_ENABLE` is enabled.

See Also

- otBackboneRouterGetState
- otBackboneRouterGetConfig
- otBackboneRouterSetConfig
- otBackboneRouterRegister

Definition at line `85` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterGetState

> otBackboneRouterState otBackboneRouterGetState (otInstance *aInstance)

Gets the Backbone Router otBackboneRouterState.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

See Also

- otBackboneRouterSetEnabled
- otBackboneRouterGetConfig
- otBackboneRouterSetConfig
- otBackboneRouterRegister

Definition at line 102 of file include/openthread/backbone_router_ftd.h

otBackboneRouterGetConfig

> void otBackboneRouterGetConfig (otInstance *aInstance, otBackboneRouterConfig *aConfig)

Gets the local Backbone Router configuration.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aConfig	A pointer where to put local Backbone Router configuration.

Available when OPENTHREAD_CONFIG_BACKBONE_ROUTER_ENABLE is enabled.

See Also

- otBackboneRouterSetEnabled
- otBackboneRouterGetState
- otBackboneRouterSetConfig
- otBackboneRouterRegister

Definition at line 119 of file include/openthread/backbone_router_ftd.h

otBackboneRouterSetConfig

> otError otBackboneRouterSetConfig (otInstance *aInstance, const otBackboneRouterConfig *aConfig)

Sets the local Backbone Router configuration otBackboneRouterConfig.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aConfig	A pointer to the Backbone Router configuration to take effect.

A Server Data Request message SRV_DATA.ntf is initiated automatically if BBR Dataset changes for Primary Backbone Router.

Available when OPENTHREAD_CONFIG_BACKBONE_ROUTER_ENABLE is enabled.

See Also

- otBackboneRouterSetEnabled
- otBackboneRouterGetState
- otBackboneRouterGetConfig
- otBackboneRouterRegister

Definition at line 141 of file include/openthread/backbone_router_ftd.h

otBackboneRouterRegister

```
otError otBackboneRouterRegister (otInstance *aInstance)
```

Explicitly registers local Backbone Router configuration.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

A Server Data Request message `SRV_DATA.ntf` is triggered for the attached device.

Available when `OPENTHREAD_CONFIG_BACKBONE_ROUTER_ENABLE` is enabled.

See Also

- otBackboneRouterSetEnabled
- otBackboneRouterGetState
- otBackboneRouterGetConfig
- otBackboneRouterSetConfig

Definition at line `161` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterGetRegistrationJitter

```
uint8_t otBackboneRouterGetRegistrationJitter (otInstance *aInstance)
```

Returns the Backbone Router registration jitter value.

Parameters

N/A	aInstance	

Returns

- The Backbone Router registration jitter value.

See Also

- otBackboneRouterSetRegistrationJitter

Definition at line `171` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterSetRegistrationJitter

```
void otBackboneRouterSetRegistrationJitter (otInstance *aInstance, uint8_t aJitter)
```

Sets the Backbone Router registration jitter value.

Parameters

[in]	aInstance	the Backbone Router registration jitter value to set.
N/A	aJitter	

See Also

- otBackboneRouterGetRegistrationJitter

Definition at line `181` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterGetDomainPrefix

```
otError otBackboneRouterGetDomainPrefix (otInstance *aInstance, otBorderRouterConfig *aConfig)
```

Gets the local Domain Prefix configuration.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aConfig	A pointer to the Domain Prefix configuration.

Definition at line `193` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterConfigNextDuaRegistrationResponse

```
void otBackboneRouterConfigNextDuaRegistrationResponse (otInstance *aInstance, const otIp6InterfaceIdentifier *aMlIid, uint8_t aStatus)
```

Configures response status for next DUA registration.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMlIid	A pointer to the Mesh Local IID. If NULL, respond with `aStatus` for any coming DUA.req, otherwise only respond the one with matching `aMlIid` .
[in]	aStatus	The status to respond.

Note: available only when `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE` is enabled. Only used for test and certification.

TODO: (DUA) support coap error code and corresponding process for certification purpose.

Definition at line `210` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterConfigNextMulticastListenerRegistrationResponse

```
void otBackboneRouterConfigNextMulticastListenerRegistrationResponse (otInstance *aInstance, uint8_t aStatus)
```

Configures the response status for the next Multicast Listener Registration.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aStatus	The status to respond.

Available when `OPENTHREAD_CONFIG_BACKBONE_ROUTER_ENABLE` , `OPENTHREAD_CONFIG_BACKBONE_ROUTER_MULTICAST_ROUTING_ENABLE` , and `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE` are enabled.

Definition at line `225` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterSetMulticastListenerCallback

```
void otBackboneRouterSetMulticastListenerCallback (otInstance *aInstance, otBackboneRouterMulticastListenerCallback aCallback, void *aContext)
```

Sets the Backbone Router Multicast Listener callback.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to the Multicast Listener callback.
[in]	aContext	A user context pointer.

Definition at line `257` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterMulticastListenerClear

```
void otBackboneRouterMulticastListenerClear (otInstance *aInstance)
```

Clears the Multicast Listeners.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Available when `OPENTHREAD_CONFIG_BACKBONE_ROUTER_ENABLE`, `OPENTHREAD_CONFIG_BACKBONE_ROUTER_MULTICAST_ROUTING_ENABLE`, and `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE` are enabled.

See Also

- otBackboneRouterMulticastListenerAdd
- otBackboneRouterMulticastListenerGetNext

Definition at line `274` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterMulticastListenerAdd

```
otError otBackboneRouterMulticastListenerAdd (otInstance *aInstance, const otIp6Address *aAddress, uint32_t aTimeout)
```

Adds a Multicast Listener with a timeout value, in seconds.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aAddress	The Multicast Listener address.
[in]	aTimeout	The timeout (in seconds) of the Multicast Listener, or 0 to use the default MLR timeout.

Pass `0` to use the default MLR timeout.

Available when `OPENTHREAD_CONFIG_BACKBONE_ROUTER_ENABLE`, `OPENTHREAD_CONFIG_BACKBONE_ROUTER_MULTICAST_ROUTING_ENABLE`, and `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE` are enabled.

See Also

- otBackboneRouterMulticastListenerClear
- otBackboneRouterMulticastListenerGetNext

Definition at line `297` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterMulticastListenerGetNext

otError otBackboneRouterMulticastListenerGetNext (otInstance *aInstance, otBackboneRouterMulticastListenerIterator *aIterator, otBackboneRouterMulticastListenerInfo *aListenerInfo)

Gets the next Multicast Listener info (using an iterator).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the iterator. On success the iterator will be updated to point to next Multicast Listener. To get the first entry the iterator should be set to OT_BACKBONE_ROUTER_MULTICAST_LISTENER_ITERATOR_INIT.
[out]	aListenerInfo	A pointer to an `otBackboneRouterMulticastListenerInfo` where information of next Multicast Listener is placed (on success).

See Also

- otBackboneRouterMulticastListenerClear
- otBackboneRouterMulticastListenerAdd

Definition at line `332` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterSetNdProxyCallback

void otBackboneRouterSetNdProxyCallback (otInstance *aInstance, otBackboneRouterNdProxyCallback aCallback, void *aContext)

Sets the Backbone Router ND Proxy callback.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to the ND Proxy callback.
[in]	aContext	A user context pointer.

Definition at line `369` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterGetNdProxyInfo

otError otBackboneRouterGetNdProxyInfo (otInstance *aInstance, const otIp6Address *aDua, otBackboneRouterNdProxyInfo *aNdProxyInfo)

Gets the Backbone Router ND Proxy info.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDua	The Domain Unicast Address.
[out]	aNdProxyInfo	A pointer to the ND Proxy info.

Definition at line `395` of file `include/openthread/backbone_router_ftd.h`

otBackboneRouterSetDomainPrefixCallback

```
void otBackboneRouterSetDomainPrefixCallback (otInstance *aInstance, otBackboneRouterDomainPrefixCallback
aCallback, void *aContext)
```

Sets the Backbone Router Domain Prefix callback.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to the Domain Prefix callback.
[in]	aContext	A user context pointer.

Definition at line 429 of file include/openthread/backbone_router_ftd.h

Macro Definition Documentation

OT_BACKBONE_ROUTER_MULTICAST_LISTENER_ITERATOR_INIT

```
#define OT_BACKBONE_ROUTER_MULTICAST_LISTENER_ITERATOR_INIT
```

Value:

```
0
```

Initializer for otBackboneRouterMulticastListenerIterator.

Definition at line 299 of file include/openthread/backbone_router_ftd.h

otBackboneRouterConfig

Represents Backbone Router configuration.

Public Attributes

uint16_t	mServer16 Only used when get Primary Backbone Router information in the Thread Network.
uint16_t	mReregistrationDelay Reregistration Delay (in seconds)
uint32_t	mMlrTimeout Multicast Listener Registration Timeout (in seconds)
uint8_t	mSequenceNumber Sequence Number.

Public Attribute Documentation

mServer16

```
uint16_t otBackboneRouterConfig::mServer16
```

Only used when get Primary Backbone Router information in the Thread Network.

Definition at line `60` of file `include/openthread/backbone_router.h`

mReregistrationDelay

```
uint16_t otBackboneRouterConfig::mReregistrationDelay
```

Reregistration Delay (in seconds)

Definition at line `61` of file `include/openthread/backbone_router.h`

mMlrTimeout

```
uint32_t otBackboneRouterConfig::mMlrTimeout
```

Multicast Listener Registration Timeout (in seconds)

Definition at line `62` of file `include/openthread/backbone_router.h`

mSequenceNumber

```
uint8_t otBackboneRouterConfig::mSequenceNumber
```

Sequence Number.

Definition at line 63 of file include/openthread/backbone_router.h

otBackboneRouterMulticastListenerInfo

Represents a Backbone Router Multicast Listener info.

Public Attributes

otIp6Address	mAddress
uint32_t	mTimeout

Public Attribute Documentation

mAddress

otIp6Address otBackboneRouterMulticastListenerInfo::mAddress

Definition at line 310 of file include/openthread/backbone_router_ftd.h

mTimeout

uint32_t otBackboneRouterMulticastListenerInfo::mTimeout

Definition at line 311 of file include/openthread/backbone_router_ftd.h

otBackboneRouterNdProxyInfo

Represents the Backbone Router ND Proxy info.

Public Attributes

otIp6InterfaceIdentifier *	mMeshLocalIid Mesh-local IID.
uint32_t	mTimeSinceLastTransaction Time since last transaction (Seconds)
uint16_t	mRloc16 RLOC16.

Public Attribute Documentation

mMeshLocalIid

```
otIp6InterfaceIdentifier* otBackboneRouterNdProxyInfo::mMeshLocalIid
```

Mesh-local IID.

Definition at line 379 of file include/openthread/backbone_router_ftd.h

mTimeSinceLastTransaction

```
uint32_t otBackboneRouterNdProxyInfo::mTimeSinceLastTransaction
```

Time since last transaction (Seconds)

Definition at line 380 of file include/openthread/backbone_router_ftd.h

mRloc16

```
uint16_t otBackboneRouterNdProxyInfo::mRloc16
```

RLOC16.

Definition at line 381 of file include/openthread/backbone_router_ftd.h

Border Agent

Border Agent

This module includes functions for the Thread Border Agent role.

Modules

otBorderAgentId

Enumerations

enum otBorderAgentState {

OT_BORDER_AGENT_STATE_STOPPED = 0
OT_BORDER_AGENT_STATE_STARTED = 1
OT_BORDER_AGENT_STATE_ACTIVE = 2

}

Defines the Border Agent state.

Typedefs

typedef struct otBorderAgentId — otBorderAgentId
Represents a Border Agent ID.

typedef enum otBorderAgentState — otBorderAgentState
Defines the Border Agent state.

Variables

OT_TOOL_PACKED_BEGIN struct otBorderAgentId — OT_TOOL_PACKED_END

Functions

otBorderAgentState — otBorderAgentGetState(otInstance *aInstance)
Gets the otBorderAgentState of the Thread Border Agent role.

uint16_t — otBorderAgentGetUdpPort(otInstance *aInstance)
Gets the UDP port of the Thread Border Agent service.

otError — otBorderAgentGetId(otInstance *aInstance, otBorderAgentId *aId)
Gets the randomly generated Border Agent ID.

otError — otBorderAgentSetId(otInstance *aInstance, const otBorderAgentId *aId)
Sets the Border Agent ID.

Macros

#define — OT_BORDER_AGENT_ID_LENGTH (16)

The length of Border Agent/Router ID in bytes.

Enumeration Documentation

otBorderAgentState

```
otBorderAgentState
```

Defines the Border Agent state.

Enumerator	
OT_BORDER_AGENT_STATE_STOPPED	Border agent role is disabled.
OT_BORDER_AGENT_STATE_STARTED	Border agent is started.
OT_BORDER_AGENT_STATE_ACTIVE	Border agent is connected with external commissioner.

Definition at line 82 of file include/openthread/border_agent.h

Typedef Documentation

otBorderAgentId

```
typedef struct otBorderAgentId otBorderAgentId
```

Represents a Border Agent ID.

Definition at line 76 of file include/openthread/border_agent.h

otBorderAgentState

```
typedef enum otBorderAgentState otBorderAgentState
```

Defines the Border Agent state.

Definition at line 87 of file include/openthread/border_agent.h

Variable Documentation

OT_TOOL_PACKED_END

```
OT_TOOL_PACKED_BEGIN struct otBorderAgentId OT_TOOL_PACKED_END
```

Definition at line 70 of file include/openthread/border_agent.h

Function Documentation

otBorderAgentGetState

```
otBorderAgentState otBorderAgentGetState (otInstance *aInstance)
```

Gets the otBorderAgentState of the Thread Border Agent role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The current otBorderAgentState of the Border Agent.

Definition at line 97 of file include/openthread/border_agent.h

otBorderAgentGetUdpPort

```
uint16_t otBorderAgentGetUdpPort (otInstance *aInstance)
```

Gets the UDP port of the Thread Border Agent service.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- UDP port of the Border Agent.

Definition at line 107 of file include/openthread/border_agent.h

otBorderAgentGetId

```
otError otBorderAgentGetId (otInstance *aInstance, otBorderAgentId *aId)
```

Gets the randomly generated Border Agent ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aId	A pointer to buffer to receive the ID.

The ID is saved in persistent storage and survives reboots. The typical use case of the ID is to be published in the MeshCoP mDNS service as the `id` TXT value for the client to identify this Border Router/Agent device.

See Also

- otBorderAgentSetId

Definition at line 125 of file include/openthread/border_agent.h

otBorderAgentSetId

```
otError otBorderAgentSetId (otInstance *aInstance, const otBorderAgentId *aId)
```

Sets the Border Agent ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[out]	aId	A pointer to the Border Agent ID.

The Border Agent ID will be saved in persistent storage and survive reboots. It's required to set the ID only once after factory reset. If the ID has never been set by calling this function, a random ID will be generated and returned when `otBorderAgentGetId` is called.

See Also

- otBorderAgentGetId

Definition at line `143` of file `include/openthread/border_agent.h`

Macro Definition Documentation

OT_BORDER_AGENT_ID_LENGTH

```
#define OT_BORDER_AGENT_ID_LENGTH
```

Value:

```
(16)
```

The length of Border Agent/Router ID in bytes.

Definition at line `58` of file `include/openthread/border_agent.h`

otBorderAgentId

Represents a Border Agent ID.

Public Attributes

uint8_t mId

Public Attribute Documentation

mId

```
uint8_t otBorderAgentId::mId[OT_BORDER_AGENT_ID_LENGTH]
```

Definition at line 69 of file include/openthread/border_agent.h

Border Router

Border Router

This module includes functions to manage local network data with the OpenThread Border Router.

Typedefs

typedef void(* otBorderRouterNetDataFullCallback)(void *aContext)
Function pointer callback which is invoked when Network Data (local or leader) gets full.

Functions

otError otBorderRouterGetNetData(otInstance *aInstance, bool aStable, uint8_t *aData, uint8_t *aDataLength)
Provides a full or stable copy of the local Thread Network Data.

otError otBorderRouterAddOnMeshPrefix(otInstance *aInstance, const otBorderRouterConfig *aConfig)
Add a border router configuration to the local network data.

otError otBorderRouterRemoveOnMeshPrefix(otInstance *aInstance, const otIp6Prefix *aPrefix)
Remove a border router configuration from the local network data.

otError otBorderRouterGetNextOnMeshPrefix(otInstance *aInstance, otNetworkDataIterator *aIterator, otBorderRouterConfig *aConfig)
Gets the next On Mesh Prefix in the local Network Data.

otError otBorderRouterAddRoute(otInstance *aInstance, const otExternalRouteConfig *aConfig)
Add an external route configuration to the local network data.

otError otBorderRouterRemoveRoute(otInstance *aInstance, const otIp6Prefix *aPrefix)
Remove an external route configuration from the local network data.

otError otBorderRouterGetNextRoute(otInstance *aInstance, otNetworkDataIterator *aIterator, otExternalRouteConfig *aConfig)
Gets the next external route in the local Network Data.

otError otBorderRouterRegister(otInstance *aInstance)
Immediately register the local network data with the Leader.

void otBorderRouterSetNetDataFullCallback(otInstance *aInstance, otBorderRouterNetDataFullCallback aCallback, void *aContext)
Sets the callback to indicate when Network Data gets full.

Typedef Documentation

otBorderRouterNetDataFullCallback

```
typedef void(* otBorderRouterNetDataFullCallback) (void *aContext) )(void *aContext)
```

Function pointer callback which is invoked when Network Data (local or leader) gets full.

Parameters

[in]	aContext	A pointer to arbitrary context information.

Definition at line 177 of file include/openthread/border_router.h

Function Documentation

otBorderRouterGetNetData

> otError otBorderRouterGetNetData (otInstance *aInstance, bool aStable, uint8_t *aData, uint8_t *aDataLength)

Provides a full or stable copy of the local Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aStable	TRUE when copying the stable version, FALSE when copying the full version.
[out]	aData	A pointer to the data buffer.
[inout]	aDataLength	On entry, size of the data buffer pointed to by aData. On exit, number of copied bytes.

Definition at line 65 of file include/openthread/border_router.h

otBorderRouterAddOnMeshPrefix

> otError otBorderRouterAddOnMeshPrefix (otInstance *aInstance, const otBorderRouterConfig *aConfig)

Add a border router configuration to the local network data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aConfig	A pointer to the border router configuration.

See Also

- otBorderRouterRemoveOnMeshPrefix
- otBorderRouterRegister

Definition at line 80 of file include/openthread/border_router.h

otBorderRouterRemoveOnMeshPrefix

> otError otBorderRouterRemoveOnMeshPrefix (otInstance *aInstance, const otIp6Prefix *aPrefix)

Remove a border router configuration from the local network data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPrefix	A pointer to an IPv6 prefix.

See Also

- otBorderRouterAddOnMeshPrefix
- otBorderRouterRegister

Definition at line 94 of file include/openthread/border_router.h

otBorderRouterGetNextOnMeshPrefix

> otError otBorderRouterGetNextOnMeshPrefix (otInstance *aInstance, otNetworkDataIterator *aIterator, otBorderRouterConfig *aConfig)

Gets the next On Mesh Prefix in the local Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the Network Data iterator context. To get the first on-mesh entry it should be set to OT_NETWORK_DATA_ITERATOR_INIT.
[out]	aConfig	A pointer to the On Mesh Prefix information.

Definition at line 108 of file include/openthread/border_router.h

otBorderRouterAddRoute

> otError otBorderRouterAddRoute (otInstance *aInstance, const otExternalRouteConfig *aConfig)

Add an external route configuration to the local network data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aConfig	A pointer to the external route configuration.

See Also

- otBorderRouterRemoveRoute
- otBorderRouterRegister

Definition at line 125 of file include/openthread/border_router.h

otBorderRouterRemoveRoute

> otError otBorderRouterRemoveRoute (otInstance *aInstance, const otIp6Prefix *aPrefix)

Remove an external route configuration from the local network data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPrefix	A pointer to an IPv6 prefix.

See Also

- otBorderRouterAddRoute
- otBorderRouterRegister

Definition at line 139 of file include/openthread/border_router.h

otBorderRouterGetNextRoute

> otError otBorderRouterGetNextRoute (otInstance *aInstance, otNetworkDataIterator *aIterator, otExternalRouteConfig *aConfig)

Gets the next external route in the local Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the Network Data iterator context. To get the first external route entry it should be set to OT_NETWORK_DATA_ITERATOR_INIT.
[out]	aConfig	A pointer to the External Route information.

Definition at line `153` of file `include/openthread/border_router.h`

otBorderRouterRegister

> otError otBorderRouterRegister (otInstance *aInstance)

Immediately register the local network data with the Leader.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

See Also

- otBorderRouterAddOnMeshPrefix
- otBorderRouterRemoveOnMeshPrefix
- otBorderRouterAddRoute
- otBorderRouterRemoveRoute

Definition at line `169` of file `include/openthread/border_router.h`

otBorderRouterSetNetDataFullCallback

> void otBorderRouterSetNetDataFullCallback (otInstance *aInstance, otBorderRouterNetDataFullCallback aCallback, void *aContext)

Sets the callback to indicate when Network Data gets full.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	The callback.
[in]	aContext	A pointer to arbitrary context information used with `aCallback`.

Requires `OPENTHREAD_CONFIG_BORDER_ROUTER_SIGNAL_NETWORK_DATA_FULL`.

The callback is invoked whenever:

- The device is acting as a leader and receives a Network Data registration from a Border Router (BR) that it cannot add to Network Data (running out of space).
- The device is acting as a BR and new entries cannot be added to its local Network Data.
- The device is acting as a BR and tries to register its local Network Data entries with the leader, but determines that its local entries will not fit.

Definition at line 196 of file include/openthread/border_router.h

Border Routing Manager

Border Routing Manager

This module includes definitions related to Border Routing Manager.

All the functions in this module require `OPENTHREAD_CONFIG_BORDER_ROUTING_ENABLE` to be enabled.

Border Routing Manager handles bi-directional routing between Thread network and adjacent infrastructure link (AIL).

It emits ICMRv6 ND Router Advertisement (RA) messages on AIL to advertise on-link and route prefixes. It also processes received RA messages from infrastructure and mirrors the discovered prefixes on the Thread Network Data to ensure devices on Thread mesh can reach AIL through the Border Router.

Routing Manager manages the Off-Mesh Routable (OMR) prefix on the Thread Network data which configures Thread devices with a suitable Off-Mesh Routable IPv6 address. It announces the reachability of this prefix on AIL by including it in the emitted RA messages as an IPv6 Route Information Option (RIO).

Routing Manager also monitors and adds on-link prefix on the infrastructure network. If a router on AIL is already providing RA messages containing an IPv6 Prefix Information Option (PIO) that enables IPv6 devices on the link to self-configure their own routable unicast IPv6 address, this address can be used by Thread devices to reach AIL. If Border Router finds no such RA message on AIL, it generates a ULA on-link prefix which it then advertises on AIL in the emitted RA messages.

Modules

otBorderRoutingPrefixTableIterator

otBorderRoutingPrefixTableEntry

Enumerations

enum otBorderRoutingState {

OT_BORDER_ROUTING_STATE_UNINITIALIZED
OT_BORDER_ROUTING_STATE_DISABLED
OT_BORDER_ROUTING_STATE_STOPPED
OT_BORDER_ROUTING_STATE_RUNNING

}

Represents the state of Border Routing Manager.

enum otBorderRoutingDhcp6PdState {

OT_BORDER_ROUTING_DHCP6_PD_STATE_DISABLED
OT_BORDER_ROUTING_DHCP6_PD_STATE_STOPPED
OT_BORDER_ROUTING_DHCP6_PD_STATE_RUNNING

}

This enumeration represents the state of DHCPv6 Prefix Delegation State.

Typedefs

typedef struct otBorderRoutingPrefixTableIterator otBorderRoutingPrefixTableIterator

Represents an iterator to iterate through the Border Router's discovered prefix table.

typedef struct otBorderRoutingPrefixTableEntry otBorderRoutingPrefixTableEntry
Represents an entry from the discovered prefix table.

Functions

otError otBorderRoutingInit(otInstance *aInstance, uint32_t aInfraIfIndex, bool aInfraIfIsRunning)
Initializes the Border Routing Manager on given infrastructure interface.

otError otBorderRoutingSetEnabled(otInstance *aInstance, bool aEnabled)
Enables or disables the Border Routing Manager.

otBorderRoutingState otBorderRoutingGetState(otInstance *aInstance)
Gets the current state of Border Routing Manager.

otRoutePreference otBorderRoutingGetRouteInfoOptionPreference(otInstance *aInstance)
Gets the current preference used when advertising Route Info Options (RIO) in Router Advertisement messages sent over the infrastructure link.

void otBorderRoutingSetRouteInfoOptionPreference(otInstance *aInstance, otRoutePreference aPreference)
Explicitly sets the preference to use when advertising Route Info Options (RIO) in Router Advertisement messages sent over the infrastructure link.

void otBorderRoutingClearRouteInfoOptionPreference(otInstance *aInstance)
Clears a previously set preference value for advertised Route Info Options.

otRoutePreference otBorderRoutingGetRoutePreference(otInstance *aInstance)
Gets the current preference used for published routes in Network Data.

void otBorderRoutingSetRoutePreference(otInstance *aInstance, otRoutePreference aPreference)
Explicitly sets the preference of published routes in Network Data.

void otBorderRoutingClearRoutePreference(otInstance *aInstance)
Clears a previously set preference value for published routes in Network Data.

otError otBorderRoutingGetOmrPrefix(otInstance *aInstance, otIp6Prefix *aPrefix)
Gets the local Off-Mesh-Routable (OMR) Prefix, for example `fdfc:1ff5:1512:5622::/64`.

otError otBorderRoutingGetPdOmrPrefix(otInstance *aInstance, otBorderRoutingPrefixTableEntry *aPrefixInfo)
Gets the DHCPv6 Prefix Delegation (PD) provided off-mesh-routable (OMR) prefix.

otError otBorderRoutingGetFavoredOmrPrefix(otInstance *aInstance, otIp6Prefix *aPrefix, otRoutePreference *aPreference)
Gets the currently favored Off-Mesh-Routable (OMR) Prefix.

otError otBorderRoutingGetOnLinkPrefix(otInstance *aInstance, otIp6Prefix *aPrefix)
Gets the local On-Link Prefix for the adjacent infrastructure link.

otError otBorderRoutingGetFavoredOnLinkPrefix(otInstance *aInstance, otIp6Prefix *aPrefix)
Gets the currently favored On-Link Prefix.

otError otBorderRoutingGetNat64Prefix(otInstance *aInstance, otIp6Prefix *aPrefix)
Gets the local NAT64 Prefix of the Border Router.

otError otBorderRoutingGetFavoredNat64Prefix(otInstance *aInstance, otIp6Prefix *aPrefix, otRoutePreference *aPreference)
Gets the currently favored NAT64 prefix.

void otBorderRoutingPrefixTableInitIterator(otInstance *aInstance, otBorderRoutingPrefixTableIterator *aIterator)
Initializes an `otBorderRoutingPrefixTableIterator`.

otError otBorderRoutingGetNextPrefixTableEntry(otInstance *aInstance, otBorderRoutingPrefixTableIterator *aIterator, otBorderRoutingPrefixTableEntry *aEntry)
Iterates over the entries in the Border Router's discovered prefix table.

void otBorderRoutingDhcp6PdSetEnabled(otInstance *aInstance, bool aEnabled)
Enables / Disables DHCPv6 Prefix Delegation.

Enumeration Documentation

otBorderRoutingState

> otBorderRoutingState

Represents the state of Border Routing Manager.

Enumerator	
OT_BORDER_ROUTING_STATE_UNINITIALIZED	Routing Manager is uninitialized.
OT_BORDER_ROUTING_STATE_DISABLED	Routing Manager is initialized but disabled.
OT_BORDER_ROUTING_STATE_STOPPED	Routing Manager in initialized and enabled but currently stopped.
OT_BORDER_ROUTING_STATE_RUNNING	Routing Manager is initialized, enabled, and running.

Definition at line 113 of file include/openthread/border_routing.h

otBorderRoutingDhcp6PdState

> otBorderRoutingDhcp6PdState

This enumeration represents the state of DHCPv6 Prefix Delegation State.

Enumerator	
OT_BORDER_ROUTING_DHCP6_PD_STATE_DISABLED	DHCPv6 PD is disabled on the border router.
OT_BORDER_ROUTING_DHCP6_PD_STATE_STOPPED	DHCPv6 PD in enabled but won't try to request and publish a prefix.
OT_BORDER_ROUTING_DHCP6_PD_STATE_RUNNING	DHCPv6 PD is enabled and will try to request and publish a prefix.

Definition at line 125 of file include/openthread/border_routing.h

Typedef Documentation

otBorderRoutingPrefixTableIterator

> typedef struct otBorderRoutingPrefixTableIterator otBorderRoutingPrefixTableIterator

Represents an iterator to iterate through the Border Router's discovered prefix table.

The fields in this type are opaque (intended for use by OpenThread core only) and therefore should not be accessed or used by caller.

Before using an iterator, it MUST be initialized using otBorderRoutingPrefixTableInitIterator() .

Definition at line 89 of file include/openthread/border_routing.h

otBorderRoutingPrefixTableEntry

typedef struct otBorderRoutingPrefixTableEntry otBorderRoutingPrefixTableEntry

Represents an entry from the discovered prefix table.

The entries in the discovered table track the Prefix/Route Info Options in the received Router Advertisement messages from other routers on infrastructure link.

Definition at line 107 of file include/openthread/border_routing.h

Function Documentation

otBorderRoutingInit

otError otBorderRoutingInit (otInstance *aInstance, uint32_t aInfraIfIndex, bool aInfraIfIsRunning)

Initializes the Border Routing Manager on given infrastructure interface.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aInfraIfIndex	The infrastructure interface index.
[in]	aInfraIfIsRunning	A boolean that indicates whether the infrastructure interface is running.

Note

- This method MUST be called before any other otBorderRouting* APIs.
- This method can be re-called to change the infrastructure interface, but the Border Routing Manager should be disabled first, and re-enabled after.

See Also

- otPlatInfraIfStateChanged.
- otBorderRoutingSetEnabled.

Definition at line 153 of file include/openthread/border_routing.h

otBorderRoutingSetEnabled

otError otBorderRoutingSetEnabled (otInstance *aInstance, bool aEnabled)

Enables or disables the Border Routing Manager.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	A boolean to enable/disable the routing manager.

Note

- The Border Routing Manager is disabled by default.

Definition at line 167 of file include/openthread/border_routing.h

otBorderRoutingGetState

```
otBorderRoutingState otBorderRoutingGetState (otInstance *aInstance)
```

Gets the current state of Border Routing Manager.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The current state of Border Routing Manager.

Definition at line `177` of file `include/openthread/border_routing.h`

otBorderRoutingGetRouteInfoOptionPreference

```
otRoutePreference otBorderRoutingGetRouteInfoOptionPreference (otInstance *aInstance)
```

Gets the current preference used when advertising Route Info Options (RIO) in Router Advertisement messages sent over the infrastructure link.

Parameters

N/A	aInstance	

The RIO preference is determined as follows:

- If explicitly set by user by calling `otBorderRoutingSetRouteInfoOptionPreference()` , the given preference is used.
- Otherwise, it is determined based on device's current role: Medium preference when in router/leader role and low preference when in child role.

Returns

- The current Route Info Option preference.

Definition at line `193` of file `include/openthread/border_routing.h`

otBorderRoutingSetRouteInfoOptionPreference

```
void otBorderRoutingSetRouteInfoOptionPreference (otInstance *aInstance, otRoutePreference aPreference)
```

Explicitly sets the preference to use when advertising Route Info Options (RIO) in Router Advertisement messages sent over the infrastructure link.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPreference	The route preference to use.

After a call to this function, BR will use the given preference for all its advertised RIOs. The preference can be cleared by calling `otBorderRoutingClearRouteInfoOptionPreference()` .

Definition at line `206` of file `include/openthread/border_routing.h`

otBorderRoutingClearRouteInfoOptionPreference

```
void otBorderRoutingClearRouteInfoOptionPreference (otInstance *aInstance)
```

Clears a previously set preference value for advertised Route Info Options.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

After a call to this function, BR will use device's role to determine the RIO preference: Medium preference when in router/leader role and low preference when in child role.

Definition at line `217` of file `include/openthread/border_routing.h`

otBorderRoutingGetRoutePreference

```
otRoutePreference otBorderRoutingGetRoutePreference (otInstance *aInstance)
```

Gets the current preference used for published routes in Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The preference is determined as follows:

- If explicitly set by user by calling `otBorderRoutingSetRoutePreference()` , the given preference is used.
- Otherwise, it is determined automatically by `RoutingManager` based on the device's role and link quality.

Returns

- The current published route preference.

Definition at line `232` of file `include/openthread/border_routing.h`

otBorderRoutingSetRoutePreference

```
void otBorderRoutingSetRoutePreference (otInstance *aInstance, otRoutePreference aPreference)
```

Explicitly sets the preference of published routes in Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPreference	The route preference to use.

After a call to this function, BR will use the given preference. The preference can be cleared by calling `otBorderRoutingClearRoutePreference()` .

Definition at line `244` of file `include/openthread/border_routing.h`

otBorderRoutingClearRoutePreference

```
void otBorderRoutingClearRoutePreference (otInstance *aInstance)
```

Clears a previously set preference value for published routes in Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

After a call to this function, BR will determine the preference automatically based on the device's role and link quality (to the parent when acting as end-device).

Definition at line `255` of file `include/openthread/border_routing.h`

otBorderRoutingGetOmrPrefix

> otError otBorderRoutingGetOmrPrefix (otInstance *aInstance, otIp6Prefix *aPrefix)

Gets the local Off-Mesh-Routable (OMR) Prefix, for example `fdfc:1ff5:1512:5622::/64`.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aPrefix	A pointer to where the prefix will be output to.

An OMR Prefix is a randomly generated 64-bit prefix that's published in the Thread network if there isn't already an OMR prefix. This prefix can be reached from the local Wi-Fi or Ethernet network.

Note: When DHCPv6 PD is enabled, the border router may publish the prefix from DHCPv6 PD.

See Also

- otBorderRoutingGetPdOmrPrefix

Definition at line `276` of file `include/openthread/border_routing.h`

otBorderRoutingGetPdOmrPrefix

> otError otBorderRoutingGetPdOmrPrefix (otInstance *aInstance, otBorderRoutingPrefixTableEntry *aPrefixInfo)

Gets the DHCPv6 Prefix Delegation (PD) provided off-mesh-routable (OMR) prefix.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aPrefixInfo	A pointer to where the prefix info will be output to.

Only mPrefix, mValidLifetime and mPreferredLifetime fields are used in the returned prefix info.

`OPENTHREAD_CONFIG_BORDER_ROUTING_DHCP6_PD_ENABLE` must be enabled.

See Also

- otBorderRoutingGetOmrPrefix
- otPlatBorderRoutingProcessIcmp6Ra

Definition at line `296` of file `include/openthread/border_routing.h`

otBorderRoutingGetFavoredOmrPrefix

> otError otBorderRoutingGetFavoredOmrPrefix (otInstance *aInstance, otIp6Prefix *aPrefix, otRoutePreference *aPreference)

Gets the currently favored Off-Mesh-Routable (OMR) Prefix.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aPrefix	A pointer to output the favored OMR prefix.
[out]	aPreference	A pointer to output the preference associated the favored prefix.

The favored OMR prefix can be discovered from Network Data or can be this device's local OMR prefix.

Definition at line `311` of file `include/openthread/border_routing.h`

otBorderRoutingGetOnLinkPrefix

> otError otBorderRoutingGetOnLinkPrefix (otInstance *aInstance, otIp6Prefix *aPrefix)

Gets the local On-Link Prefix for the adjacent infrastructure link.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aPrefix	A pointer to where the prefix will be output to.

The local On-Link Prefix is a 64-bit prefix that's advertised on the infrastructure link if there isn't already a usable on-link prefix being advertised on the link.

Definition at line `326` of file `include/openthread/border_routing.h`

otBorderRoutingGetFavoredOnLinkPrefix

> otError otBorderRoutingGetFavoredOnLinkPrefix (otInstance *aInstance, otIp6Prefix *aPrefix)

Gets the currently favored On-Link Prefix.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aPrefix	A pointer to where the prefix will be output to.

The favored prefix is either a discovered on-link prefix on the infrastructure link or the local on-link prefix.

Definition at line `340` of file `include/openthread/border_routing.h`

otBorderRoutingGetNat64Prefix

> otError otBorderRoutingGetNat64Prefix (otInstance *aInstance, otIp6Prefix *aPrefix)

Gets the local NAT64 Prefix of the Border Router.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aPrefix	A pointer to where the prefix will be output to.

NAT64 Prefix might not be advertised in the Thread network.

`OPENTHREAD_CONFIG_NAT64_BORDER_ROUTING_ENABLE` must be enabled.

Definition at line `356` of file `include/openthread/border_routing.h`

otBorderRoutingGetFavoredNat64Prefix

```
otError otBorderRoutingGetFavoredNat64Prefix (otInstance *aInstance, otIp6Prefix *aPrefix, otRoutePreference *aPreference)
```

Gets the currently favored NAT64 prefix.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aPrefix	A pointer to output the favored NAT64 prefix.
[out]	aPreference	A pointer to output the preference associated the favored prefix.

The favored NAT64 prefix can be discovered from infrastructure link or can be this device's local NAT64 prefix.

Definition at line `371` of file `include/openthread/border_routing.h`

otBorderRoutingPrefixTableInitIterator

```
void otBorderRoutingPrefixTableInitIterator (otInstance *aInstance, otBorderRoutingPrefixTableIterator *aIterator)
```

Initializes an `otBorderRoutingPrefixTableIterator`.

Parameters

[in]	aInstance	The OpenThread instance.
[out]	aIterator	A pointer to the iterator to initialize.

An iterator MUST be initialized before it is used.

An iterator can be initialized again to restart from the beginning of the table.

When iterating over entries in the table, to ensure the update times `mMsecSinceLastUpdate` of entries are consistent, they are given relative to the time the iterator was initialized.

Definition at line `389` of file `include/openthread/border_routing.h`

otBorderRoutingGetNextPrefixTableEntry

```
otError otBorderRoutingGetNextPrefixTableEntry (otInstance *aInstance, otBorderRoutingPrefixTableIterator *aIterator, otBorderRoutingPrefixTableEntry *aEntry)
```

Iterates over the entries in the Border Router's discovered prefix table.

Parameters

[in]	aInstance	The OpenThread instance.
[inout]	aIterator	A pointer to the iterator.
[out]	aEntry	A pointer to the entry to populate.

Definition at line `402` of file `include/openthread/border_routing.h`

otBorderRoutingDhcp6PdSetEnabled

```
void otBorderRoutingDhcp6PdSetEnabled (otInstance *aInstance, bool aEnabled)
```

Enables / Disables DHCPv6 Prefix Delegation.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	Whether to accept platform generated RA messages.

`OPENTHREAD_CONFIG_BORDER_ROUTING_DHCP6_PD_ENABLE` must be enabled.

Definition at line `415` of file `include/openthread/border_routing.h`

otBorderRoutingPrefixTableIterator

Represents an iterator to iterate through the Border Router's discovered prefix table.

The fields in this type are opaque (intended for use by OpenThread core only) and therefore should not be accessed or used by caller.

Before using an iterator, it MUST be initialized using `otBorderRoutingPrefixTableInitIterator()` .

Public Attributes

const void *	mPtr1
const void *	mPtr2
uint32_t	mData32

Public Attribute Documentation

mPtr1

```
const void* otBorderRoutingPrefixTableIterator::mPtr1
```

Definition at line `86` of file `include/openthread/border_routing.h`

mPtr2

```
const void* otBorderRoutingPrefixTableIterator::mPtr2
```

Definition at line `87` of file `include/openthread/border_routing.h`

mData32

```
uint32_t otBorderRoutingPrefixTableIterator::mData32
```

Definition at line `88` of file `include/openthread/border_routing.h`

otBorderRoutingPrefixTableEntry

Represents an entry from the discovered prefix table.

The entries in the discovered table track the Prefix/Route Info Options in the received Router Advertisement messages from other routers on infrastructure link.

Public Attributes

otIp6Address	mRouterAddress IPv6 address of the router.
otIp6Prefix	mPrefix The discovered IPv6 prefix.
bool	mIsOnLink Indicates whether the prefix is on-link or route prefix.
uint32_t	mMsecSinceLastUpdate Milliseconds since last update of this prefix.
uint32_t	mValidLifetime Valid lifetime of the prefix (in seconds).
otRoutePreference	mRoutePreference Route preference when `mIsOnlink` is false.
uint32_t	mPreferredLifetime Preferred lifetime of the on-link prefix when `mIsOnLink` is true.

Public Attribute Documentation

mRouterAddress

```
otIp6Address otBorderRoutingPrefixTableEntry::mRouterAddress
```

IPv6 address of the router.

Definition at line `100` of file `include/openthread/border_routing.h`

mPrefix

```
otIp6Prefix otBorderRoutingPrefixTableEntry::mPrefix
```

The discovered IPv6 prefix.

Definition at line `101` of file `include/openthread/border_routing.h`

mIsOnLink

```
bool otBorderRoutingPrefixTableEntry::mIsOnLink
```

Indicates whether the prefix is on-link or route prefix.

Definition at line `102` of file `include/openthread/border_routing.h`

mMsecSinceLastUpdate

```
uint32_t otBorderRoutingPrefixTableEntry::mMsecSinceLastUpdate
```

Milliseconds since last update of this prefix.

Definition at line `103` of file `include/openthread/border_routing.h`

mValidLifetime

```
uint32_t otBorderRoutingPrefixTableEntry::mValidLifetime
```

Valid lifetime of the prefix (in seconds).

Definition at line `104` of file `include/openthread/border_routing.h`

mRoutePreference

```
otRoutePreference otBorderRoutingPrefixTableEntry::mRoutePreference
```

Route preference when `mIsOnlink` is false.

Definition at line `105` of file `include/openthread/border_routing.h`

mPreferredLifetime

```
uint32_t otBorderRoutingPrefixTableEntry::mPreferredLifetime
```

Preferred lifetime of the on-link prefix when `mIsOnLink` is true.

Definition at line `106` of file `include/openthread/border_routing.h`

Commissioner

Commissioner

This module includes functions for the Thread Commissioner role.

Modules

otSteeringData

otCommissioningDataset

otJoinerPskd

otJoinerInfo

Enumerations

enum otCommissionerState {

OT_COMMISSIONER_STATE_DISABLED = 0
OT_COMMISSIONER_STATE_PETITION = 1
OT_COMMISSIONER_STATE_ACTIVE = 2

}

Defines the Commissioner State.

enum otCommissionerJoinerEvent {

OT_COMMISSIONER_JOINER_START = 0
OT_COMMISSIONER_JOINER_CONNECTED = 1
OT_COMMISSIONER_JOINER_FINALIZE = 2
OT_COMMISSIONER_JOINER_END = 3
OT_COMMISSIONER_JOINER_REMOVED = 4

}

Defines a Joiner Event on the Commissioner.

enum otJoinerInfoType {

OT_JOINER_INFO_TYPE_ANY = 0
OT_JOINER_INFO_TYPE_EUI64 = 1
OT_JOINER_INFO_TYPE_DISCERNER = 2

}

Defines a Joiner Info Type.

Typedefs

typedef enum otCommissionerState
Defines the Commissioner State.

typedef enum otCommissionerJoinerEvent
Defines a Joiner Event on the Commissioner.

typedef struct otSteeringData
Represents the steering data.

typedef struct otCommissioningDataset
Represents a Commissioning Dataset.

typedef struct otJoinerPskd
Represents a Joiner PSKd.

typedef enum otJoinerInfoType
Defines a Joiner Info Type.

typedef struct otJoinerInfo
Represents a Joiner Info.

typedef void(* otCommissionerStateCallback)(otCommissionerState aState, void *aContext)
Pointer is called whenever the commissioner state changes.

typedef void(* otCommissionerJoinerCallback)(otCommissionerJoinerEvent aEvent, const otJoinerInfo *aJoinerInfo, const otExtAddress *aJoinerId, void *aContext)
Pointer is called whenever the joiner state changes.

typedef void(* otCommissionerEnergyReportCallback)(uint32_t aChannelMask, const uint8_t *aEnergyList, uint8_t aEnergyListLength, void *aContext)
Pointer is called when the Commissioner receives an Energy Report.

typedef void(* otCommissionerPanIdConflictCallback)(uint16_t aPanId, uint32_t aChannelMask, void *aContext)
Pointer is called when the Commissioner receives a PAN ID Conflict message.

Functions

otError otCommissionerStart(otInstance *aInstance, otCommissionerStateCallback aStateCallback, otCommissionerJoinerCallback aJoinerCallback, void *aCallbackContext)
Enables the Thread Commissioner role.

otError otCommissionerStop(otInstance *aInstance)
Disables the Thread Commissioner role.

const char * otCommissionerGetId(otInstance *aInstance)
Returns the Commissioner Id.

otError otCommissionerSetId(otInstance *aInstance, const char *aId)
Sets the Commissioner Id.

otError otCommissionerAddJoiner(otInstance *aInstance, const otExtAddress *aEui64, const char *aPskd, uint32_t aTimeout)
Adds a Joiner entry.

otError otCommissionerAddJoinerWithDiscerner(otInstance *aInstance, const otJoinerDiscerner *aDiscerner, const char *aPskd, uint32_t aTimeout)
Adds a Joiner entry with a given Joiner Discerner value.

otError otCommissionerGetNextJoinerInfo(otInstance *aInstance, uint16_t *aIterator, otJoinerInfo *aJoiner)
Get joiner info at iterator position.

otError otCommissionerRemoveJoiner(otInstance *aInstance, const otExtAddress *aEui64)
Removes a Joiner entry.

otError otCommissionerRemoveJoinerWithDiscerner(otInstance *aInstance, const otJoinerDiscerner *aDiscerner)
Removes a Joiner entry.

const char * otCommissionerGetProvisioningUrl(otInstance *aInstance)
Gets the Provisioning URL.

otError	otCommissionerSetProvisioningUrl(otInstance *aInstance, const char *aProvisioningUrl) Sets the Provisioning URL.
otError	otCommissionerAnnounceBegin(otInstance *aInstance, uint32_t aChannelMask, uint8_t aCount, uint16_t aPeriod, const otIp6Address *aAddress) Sends an Announce Begin message.
otError	otCommissionerEnergyScan(otInstance *aInstance, uint32_t aChannelMask, uint8_t aCount, uint16_t aPeriod, uint16_t aScanDuration, const otIp6Address *aAddress, otCommissionerEnergyReportCallback aCallback, void *aContext) Sends an Energy Scan Query message.
otError	otCommissionerPanIdQuery(otInstance *aInstance, uint16_t aPanId, uint32_t aChannelMask, const otIp6Address *aAddress, otCommissionerPanIdConflictCallback aCallback, void *aContext) Sends a PAN ID Query message.
otError	otCommissionerSendMgmtGet(otInstance *aInstance, const uint8_t *aTlvs, uint8_t aLength) Sends MGMT_COMMISSIONER_GET.
otError	otCommissionerSendMgmtSet(otInstance *aInstance, const otCommissioningDataset *aDataset, const uint8_t *aTlvs, uint8_t aLength) Sends MGMT_COMMISSIONER_SET.
uint16_t	otCommissionerGetSessionId(otInstance *aInstance) Returns the Commissioner Session ID.
otCommissionerState	otCommissionerGetState(otInstance *aInstance) Returns the Commissioner State.

Macros

#define	OT_COMMISSIONING_PASSPHRASE_MIN_SIZE 6 Minimum size of the Commissioning Passphrase.
#define	OT_COMMISSIONING_PASSPHRASE_MAX_SIZE 255 Maximum size of the Commissioning Passphrase.
#define	OT_PROVISIONING_URL_MAX_SIZE 64 Max size (number of chars) in Provisioning URL string (excludes null char).
#define	OT_STEERING_DATA_MAX_LENGTH 16 Max steering data length (bytes)
#define	OT_JOINER_MAX_PSKD_LENGTH 32 Maximum string length of a Joiner PSKd (does not include null char).

Enumeration Documentation

otCommissionerState

> otCommissionerState

Defines the Commissioner State.

	Enumerator
OT_COMMISSIONER_STATE_DISABLED	Commissioner role is disabled.
OT_COMMISSIONER_STATE_PETITION	Currently petitioning to become a Commissioner.
OT_COMMISSIONER_STATE_ACTIVE	Commissioner role is active.

Definition at line 62 of file include/openthread/commissioner.h

otCommissionerJoinerEvent

otCommissionerJoinerEvent

Defines a Joiner Event on the Commissioner.

Enumerator	
OT_COMMISSIONER_JOINER_START	
OT_COMMISSIONER_JOINER_CONNECTED	
OT_COMMISSIONER_JOINER_FINALIZE	
OT_COMMISSIONER_JOINER_END	
OT_COMMISSIONER_JOINER_REMOVED	

Definition at line 73 of file include/openthread/commissioner.h

otJoinerInfoType

otJoinerInfoType

Defines a Joiner Info Type.

Enumerator	
OT_JOINER_INFO_TYPE_ANY	Accept any Joiner (no EUI64 or Discerner is specified).
OT_JOINER_INFO_TYPE_EUI64	Joiner EUI-64 is specified (mSharedId.mEui64 in otJoinerInfo).
OT_JOINER_INFO_TYPE_DISCERNER	Joiner Discerner is specified (mSharedId.mDiscerner in otJoinerInfo).

Definition at line 132 of file include/openthread/commissioner.h

Typedef Documentation

otCommissionerState

typedef enum otCommissionerState otCommissionerState

Defines the Commissioner State.

Definition at line 67 of file include/openthread/commissioner.h

otCommissionerJoinerEvent

typedef enum otCommissionerJoinerEvent otCommissionerJoinerEvent

Defines a Joiner Event on the Commissioner.

Definition at line 80 of file include/openthread/commissioner.h

otSteeringData

typedef struct otSteeringData otSteeringData

Represents the steering data.

Definition at line 97 of file include/openthread/commissioner.h

otCommissioningDataset

```
typedef struct otCommissioningDataset otCommissioningDataset
```

Represents a Commissioning Dataset.

Definition at line 115 of file include/openthread/commissioner.h

otJoinerPskd

```
typedef struct otJoinerPskd otJoinerPskd
```

Represents a Joiner PSKd.

Definition at line 126 of file include/openthread/commissioner.h

otJoinerInfoType

```
typedef enum otJoinerInfoType otJoinerInfoType
```

Defines a Joiner Info Type.

Definition at line 137 of file include/openthread/commissioner.h

otJoinerInfo

```
typedef struct otJoinerInfo otJoinerInfo
```

Represents a Joiner Info.

Definition at line 153 of file include/openthread/commissioner.h

otCommissionerStateCallback

```
typedef void(* otCommissionerStateCallback) (otCommissionerState aState, void *aContext) )(otCommissionerState aState, void *aContext)
```

Pointer is called whenever the commissioner state changes.

Parameters

[in]	aState	The Commissioner state.
[in]	aContext	A pointer to application-specific context.

Definition at line 162 of file include/openthread/commissioner.h

otCommissionerJoinerCallback

```
typedef void(* otCommissionerJoinerCallback) (otCommissionerJoinerEvent aEvent, const otJoinerInfo *aJoinerInfo, const otExtAddress *aJoinerId, void *aContext) )(otCommissionerJoinerEvent aEvent, const otJoinerInfo *aJoinerInfo, const otExtAddress *aJoinerId, void *aContext)
```

Pointer is called whenever the joiner state changes.

Parameters

[in]	aEvent	The joiner event type.
[in]	aJoinerInfo	A pointer to the Joiner Info.
[in]	aJoinerId	A pointer to the Joiner ID (if not known, it will be NULL).
[in]	aContext	A pointer to application-specific context.

Definition at line `173` of file `include/openthread/commissioner.h`

otCommissionerEnergyReportCallback

```
typedef void(* otCommissionerEnergyReportCallback) (uint32_t aChannelMask, const uint8_t *aEnergyList, uint8_t aEnergyListLength, void *aContext) )(uint32_t aChannelMask, const uint8_t *aEnergyList, uint8_t aEnergyListLength, void *aContext)
```

Pointer is called when the Commissioner receives an Energy Report.

Parameters

[in]	aChannelMask	The channel mask value.
[in]	aEnergyList	A pointer to the energy measurement list.
[in]	aEnergyListLength	Number of entries in `aEnergyListLength`.
[in]	aContext	A pointer to application-specific context.

Definition at line `370` of file `include/openthread/commissioner.h`

otCommissionerPanIdConflictCallback

```
typedef void(* otCommissionerPanIdConflictCallback) (uint16_t aPanId, uint32_t aChannelMask, void *aContext) )(uint16_t aPanId, uint32_t aChannelMask, void *aContext)
```

Pointer is called when the Commissioner receives a PAN ID Conflict message.

Parameters

[in]	aPanId	The PAN ID value.
[in]	aChannelMask	The channel mask value.
[in]	aContext	A pointer to application-specific context.

Definition at line `411` of file `include/openthread/commissioner.h`

Function Documentation

otCommissionerStart

> otError otCommissionerStart (otInstance *aInstance, otCommissionerStateCallback aStateCallback, otCommissionerJoinerCallback aJoinerCallback, void *aCallbackContext)

Enables the Thread Commissioner role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aStateCallback	A pointer to a function that is called when the commissioner state changes.
[in]	aJoinerCallback	A pointer to a function that is called with a joiner event occurs.
[in]	aCallbackContext	A pointer to application-specific context.

Definition at line `191` of file `include/openthread/commissioner.h`

otCommissionerStop

> otError otCommissionerStop (otInstance *aInstance)

Disables the Thread Commissioner role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `205` of file `include/openthread/commissioner.h`

otCommissionerGetId

> const char * otCommissionerGetId (otInstance *aInstance)

Returns the Commissioner Id.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Commissioner Id.

Definition at line `215` of file `include/openthread/commissioner.h`

otCommissionerSetId

> otError otCommissionerSetId (otInstance *aInstance, const char *aId)

Sets the Commissioner Id.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aId	A pointer to a string character array. Must be null terminated.

Definition at line `228` of file `include/openthread/commissioner.h`

otCommissionerAddJoiner

> otError otCommissionerAddJoiner (otInstance *aInstance, const otExtAddress *aEui64, const char *aPskd, uint32_t aTimeout)

Adds a Joiner entry.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEui64	A pointer to the Joiner's IEEE EUI-64 or NULL for any Joiner.
[in]	aPskd	A pointer to the PSKd.
[in]	aTimeout	A time after which a Joiner is automatically removed, in seconds.

Note

- Only use this after successfully starting the Commissioner role with otCommissionerStart().

Definition at line `246` of file `include/openthread/commissioner.h`

otCommissionerAddJoinerWithDiscerner

> otError otCommissionerAddJoinerWithDiscerner (otInstance *aInstance, const otJoinerDiscerner *aDiscerner, const char *aPskd, uint32_t aTimeout)

Adds a Joiner entry with a given Joiner Discerner value.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDiscerner	A pointer to the Joiner Discerner.
[in]	aPskd	A pointer to the PSKd.
[in]	aTimeout	A time after which a Joiner is automatically removed, in seconds.

Note

- Only use this after successfully starting the Commissioner role with otCommissionerStart().

Definition at line `267` of file `include/openthread/commissioner.h`

otCommissionerGetNextJoinerInfo

> otError otCommissionerGetNextJoinerInfo (otInstance *aInstance, uint16_t *aIterator, otJoinerInfo *aJoiner)

Get joiner info at iterator position.

Parameters

[in]	aInstance	A pointer to instance.
[inout]	aIterator	A pointer to the Joiner Info iterator context.
[out]	aJoiner	A reference to Joiner info.

Definition at line `283` of file `include/openthread/commissioner.h`

otCommissionerRemoveJoiner

```
otError otCommissionerRemoveJoiner (otInstance *aInstance, const otExtAddress *aEui64)
```

Removes a Joiner entry.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEui64	A pointer to the Joiner's IEEE EUI-64 or NULL for any Joiner.

Note

- Only use this after successfully starting the Commissioner role with otCommissionerStart().

Definition at line 299 of file include/openthread/commissioner.h

otCommissionerRemoveJoinerWithDiscerner

```
otError otCommissionerRemoveJoinerWithDiscerner (otInstance *aInstance, const otJoinerDiscerner *aDiscerner)
```

Removes a Joiner entry.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDiscerner	A pointer to the Joiner Discerner.

Note

- Only use this after successfully starting the Commissioner role with otCommissionerStart().

Definition at line 315 of file include/openthread/commissioner.h

otCommissionerGetProvisioningUrl

```
const char * otCommissionerGetProvisioningUrl (otInstance *aInstance)
```

Gets the Provisioning URL.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the URL string.

Definition at line 325 of file include/openthread/commissioner.h

otCommissionerSetProvisioningUrl

```
otError otCommissionerSetProvisioningUrl (otInstance *aInstance, const char *aProvisioningUrl)
```

Sets the Provisioning URL.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aProvisioningUrl	A pointer to the Provisioning URL (may be NULL to set as empty string).

Definition at line 337 of file include/openthread/commissioner.h

otCommissionerAnnounceBegin

> otError otCommissionerAnnounceBegin (otInstance *aInstance, uint32_t aChannelMask, uint8_t aCount, uint16_t aPeriod, const otIp6Address *aAddress)

Sends an Announce Begin message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChannelMask	The channel mask value.
[in]	aCount	The number of Announcement messages per channel.
[in]	aPeriod	The time between two successive MLE Announce transmissions (in milliseconds).
[in]	aAddress	A pointer to the IPv6 destination.

Note

- Only use this after successfully starting the Commissioner role with otCommissionerStart().

Definition at line 355 of file include/openthread/commissioner.h

otCommissionerEnergyScan

> otError otCommissionerEnergyScan (otInstance *aInstance, uint32_t aChannelMask, uint8_t aCount, uint16_t aPeriod, uint16_t aScanDuration, const otIp6Address *aAddress, otCommissionerEnergyReportCallback aCallback, void *aContext)

Sends an Energy Scan Query message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChannelMask	The channel mask value.
[in]	aCount	The number of energy measurements per channel.
[in]	aPeriod	The time between energy measurements (milliseconds).
[in]	aScanDuration	The scan duration for each energy measurement (milliseconds).
[in]	aAddress	A pointer to the IPv6 destination.
[in]	aCallback	A pointer to a function called on receiving an Energy Report message.
[in]	aContext	A pointer to application-specific context.

Note

- Only use this after successfully starting the Commissioner role with otCommissionerStart().

Definition at line 394 of file include/openthread/commissioner.h

otCommissionerPanIdQuery

> otError otCommissionerPanIdQuery (otInstance *aInstance, uint16_t aPanId, uint32_t aChannelMask, const otIp6Address *aAddress, otCommissionerPanIdConflictCallback aCallback, void *aContext)

Sends a PAN ID Query message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPanId	The PAN ID to query.
[in]	aChannelMask	The channel mask value.
[in]	aAddress	A pointer to the IPv6 destination.
[in]	aCallback	A pointer to a function called on receiving a PAN ID Conflict message.
[in]	aContext	A pointer to application-specific context.

Note

- Only use this after successfully starting the Commissioner role with otCommissionerStart().

Definition at line `430` of file `include/openthread/commissioner.h`

otCommissionerSendMgmtGet

> otError otCommissionerSendMgmtGet (otInstance *aInstance, const uint8_t *aTlvs, uint8_t aLength)

Sends MGMT_COMMISSIONER_GET.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aTlvs	A pointer to TLVs.
[in]	aLength	The length of TLVs.

Definition at line `449` of file `include/openthread/commissioner.h`

otCommissionerSendMgmtSet

> otError otCommissionerSendMgmtSet (otInstance *aInstance, const otCommissioningDataset *aDataset, const uint8_t *aTlvs, uint8_t aLength)

Sends MGMT_COMMISSIONER_SET.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDataset	A pointer to commissioning dataset.
[in]	aTlvs	A pointer to TLVs.
[in]	aLength	The length of TLVs.

Definition at line `464` of file `include/openthread/commissioner.h`

otCommissionerGetSessionId

```
uint16_t otCommissionerGetSessionId (otInstance *aInstance)
```

Returns the Commissioner Session ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The current commissioner session id.

Definition at line `477` of file `include/openthread/commissioner.h`

otCommissionerGetState

```
otCommissionerState otCommissionerGetState (otInstance *aInstance)
```

Returns the Commissioner State.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `489` of file `include/openthread/commissioner.h`

Macro Definition Documentation

OT_COMMISSIONING_PASSPHRASE_MIN_SIZE

```
#define OT_COMMISSIONING_PASSPHRASE_MIN_SIZE
```

Value:

```
6
```

Minimum size of the Commissioning Passphrase.

Definition at line `82` of file `include/openthread/commissioner.h`

OT_COMMISSIONING_PASSPHRASE_MAX_SIZE

```
#define OT_COMMISSIONING_PASSPHRASE_MAX_SIZE
```

Value:

```
255
```

Maximum size of the Commissioning Passphrase.

Definition at line `83` of file `include/openthread/commissioner.h`

OT_PROVISIONING_URL_MAX_SIZE

```
#define OT_PROVISIONING_URL_MAX_SIZE
```

Value:

```
64
```

Max size (number of chars) in Provisioning URL string (excludes null char).

Definition at line 85 of file include/openthread/commissioner.h

OT_STEERING_DATA_MAX_LENGTH

```
#define OT_STEERING_DATA_MAX_LENGTH
```

Value:

```
16
```

Max steering data length (bytes)

Definition at line 87 of file include/openthread/commissioner.h

OT_JOINER_MAX_PSKD_LENGTH

```
#define OT_JOINER_MAX_PSKD_LENGTH
```

Value:

```
32
```

Maximum string length of a Joiner PSKd (does not include null char).

Definition at line 117 of file include/openthread/commissioner.h

otSteeringData

Represents the steering data.

Public Attributes

uint8_t	mLength Length of steering data (bytes)
uint8_t	m8 Byte values.

Public Attribute Documentation

mLength

```
uint8_t otSteeringData::mLength
```

Length of steering data (bytes)

Definition at line 95 of file include/openthread/commissioner.h

m8

```
uint8_t otSteeringData::m8[OT_STEERING_DATA_MAX_LENGTH]
```

Byte values.

Definition at line 96 of file include/openthread/commissioner.h

otCommissioningDataset

Represents a Commissioning Dataset.

Public Attributes

uint16_t	mLocator Border Router RLOC16.
uint16_t	mSessionId Commissioner Session Id.
otSteeringData	mSteeringData Steering Data.
uint16_t	mJoinerUdpPort Joiner UDP Port.
bool	mIsLocatorSet TRUE if Border Router RLOC16 is set, FALSE otherwise.
bool	mIsSessionIdSet TRUE if Commissioner Session Id is set, FALSE otherwise.
bool	mIsSteeringDataSet TRUE if Steering Data is set, FALSE otherwise.
bool	mIsJoinerUdpPortSet TRUE if Joiner UDP Port is set, FALSE otherwise.
bool	mHasExtraTlv TRUE if the Dataset contains any extra unknown sub-TLV, FALSE otherwise.

Public Attribute Documentation

mLocator

```
uint16_t otCommissioningDataset::mLocator
```

Border Router RLOC16.

Definition at line `105` of file `include/openthread/commissioner.h`

mSessionId

```
uint16_t otCommissioningDataset::mSessionId
```

Commissioner Session Id.

Definition at line `106` of file `include/openthread/commissioner.h`

mSteeringData

```
otSteeringData otCommissioningDataset::mSteeringData
```

Steering Data.

Definition at line 107 of file include/openthread/commissioner.h

mJoinerUdpPort

```
uint16_t otCommissioningDataset::mJoinerUdpPort
```

Joiner UDP Port.

Definition at line 108 of file include/openthread/commissioner.h

mIsLocatorSet

```
bool otCommissioningDataset::mIsLocatorSet
```

TRUE if Border Router RLOC16 is set, FALSE otherwise.

Definition at line 110 of file include/openthread/commissioner.h

mIsSessionIdSet

```
bool otCommissioningDataset::mIsSessionIdSet
```

TRUE if Commissioner Session Id is set, FALSE otherwise.

Definition at line 111 of file include/openthread/commissioner.h

mIsSteeringDataSet

```
bool otCommissioningDataset::mIsSteeringDataSet
```

TRUE if Steering Data is set, FALSE otherwise.

Definition at line 112 of file include/openthread/commissioner.h

mIsJoinerUdpPortSet

```
bool otCommissioningDataset::mIsJoinerUdpPortSet
```

TRUE if Joiner UDP Port is set, FALSE otherwise.

Definition at line 113 of file include/openthread/commissioner.h

mHasExtraTlv

bool otCommissioningDataset::mHasExtraTlv

TRUE if the Dataset contains any extra unknown sub-TLV, FALSE otherwise.

Definition at line 114 of file include/openthread/commissioner.h

otJoinerPskd

Represents a Joiner PSKd.

Public Attributes

char	m8 Char string array (must be null terminated - +1 is for null char).

Public Attribute Documentation

m8

```
char otJoinerPskd::m8[OT_JOINER_MAX_PSKD_LENGTH+1]
```

Char string array (must be null terminated - +1 is for null char).

Definition at line 125 of file include/openthread/commissioner.h

otJoinerInfo

Represents a Joiner Info.

Public Attributes

otJoinerInfoType	mType Joiner type.
otExtAddress	mEui64 Joiner EUI64 (when `mType` is `OT_JOINER_INFO_TYPE_EUI64`)
otJoinerDiscerner	mDiscerner Joiner Discerner (when `mType` is `OT_JOINER_INFO_TYPE_DISCERNER`)
union otJoinerInfo::@0	mSharedId Shared fields.
otJoinerPskd	mPskd Joiner PSKd.
uint32_t	mExpirationTime Joiner expiration time in msec.

Public Attribute Documentation

mType

```
otJoinerInfoType otJoinerInfo::mType
```

Joiner type.

Definition at line `145` of file `include/openthread/commissioner.h`

mEui64

```
otExtAddress otJoinerInfo::mEui64
```

Joiner EUI64 (when `mType` is `OT_JOINER_INFO_TYPE_EUI64`)

Definition at line `148` of file `include/openthread/commissioner.h`

mDiscerner

```
otJoinerDiscerner otJoinerInfo::mDiscerner
```

Joiner Discerner (when `mType` is `OT_JOINER_INFO_TYPE_DISCERNER`)

Definition at line `149` of file `include/openthread/commissioner.h`

> union otJoinerInfo::@0 otJoinerInfo::mSharedId

Shared fields.

Definition at line 150 of file include/openthread/commissioner.h

mPskd

> otJoinerPskd otJoinerInfo::mPskd

Joiner PSKd.

Definition at line 151 of file include/openthread/commissioner.h

mExpirationTime

> uint32_t otJoinerInfo::mExpirationTime

Joiner expiration time in msec.

Definition at line 152 of file include/openthread/commissioner.h

General

General

This module includes functions for all Thread roles.

The Network Data Publisher provides mechanisms to limit the number of similar Service and/or Prefix (on-mesh prefix or external route) entries in the Thread Network Data by monitoring the Network Data and managing if or when to add or remove entries.

All the functions in this module require `OPENTHREAD_CONFIG_NETDATA_PUBLISHER_ENABLE` to be enabled.

Note

- The functions in this module require `OPENTHREAD_FTD=1` or `OPENTHREAD_MTD=1`.

Modules

otBorderRouterConfig

otLowpanContextInfo

otExternalRouteConfig

otServerConfig

otServiceConfig

otNetworkDiagConnectivity

otNetworkDiagRouteData

otNetworkDiagRoute

otNetworkDiagMacCounters

otNetworkDiagMleCounters

otNetworkDiagChildEntry

otNetworkDiagTlv

otLinkModeConfig

otNeighborInfo

otLeaderData

otRouterInfo

otIpCounters

otMleCounters

otThreadParentResponseInfo

otThreadDiscoveryRequestInfo

Enumerations

enum otRoutePreference {

OT_ROUTE_PREFERENCE_LOW = -1
OT_ROUTE_PREFERENCE_MED = 0
OT_ROUTE_PREFERENCE_HIGH = 1

}

Defines valid values for `mPreference` in `otExternalRouteConfig` and `otBorderRouterConfig`.

enum otNetDataPublisherEvent {

OT_NETDATA_PUBLISHER_EVENT_ENTRY_ADDED = 0
OT_NETDATA_PUBLISHER_EVENT_ENTRY_REMOVED = 1

}

Represents the events reported from the Publisher callbacks.

enum @5 {

OT_NETWORK_DIAGNOSTIC_TLV_EXT_ADDRESS = 0
OT_NETWORK_DIAGNOSTIC_TLV_SHORT_ADDRESS = 1
OT_NETWORK_DIAGNOSTIC_TLV_MODE = 2
OT_NETWORK_DIAGNOSTIC_TLV_TIMEOUT = 3
OT_NETWORK_DIAGNOSTIC_TLV_CONNECTIVITY = 4
OT_NETWORK_DIAGNOSTIC_TLV_ROUTE = 5
OT_NETWORK_DIAGNOSTIC_TLV_LEADER_DATA = 6
OT_NETWORK_DIAGNOSTIC_TLV_NETWORK_DATA = 7
OT_NETWORK_DIAGNOSTIC_TLV_IP6_ADDR_LIST = 8
OT_NETWORK_DIAGNOSTIC_TLV_MAC_COUNTERS = 9
OT_NETWORK_DIAGNOSTIC_TLV_BATTERY_LEVEL = 14
OT_NETWORK_DIAGNOSTIC_TLV_SUPPLY_VOLTAGE = 15
OT_NETWORK_DIAGNOSTIC_TLV_CHILD_TABLE = 16
OT_NETWORK_DIAGNOSTIC_TLV_CHANNEL_PAGES = 17
OT_NETWORK_DIAGNOSTIC_TLV_TYPE_LIST = 18
OT_NETWORK_DIAGNOSTIC_TLV_MAX_CHILD_TIMEOUT = 19
OT_NETWORK_DIAGNOSTIC_TLV_VERSION = 24
OT_NETWORK_DIAGNOSTIC_TLV_VENDOR_NAME = 25
OT_NETWORK_DIAGNOSTIC_TLV_VENDOR_MODEL = 26
OT_NETWORK_DIAGNOSTIC_TLV_VENDOR_SW_VERSION = 27
OT_NETWORK_DIAGNOSTIC_TLV_THREAD_STACK_VERSION = 28
OT_NETWORK_DIAGNOSTIC_TLV_CHILD = 29
OT_NETWORK_DIAGNOSTIC_TLV_CHILD_IP6_ADDR_LIST = 30
OT_NETWORK_DIAGNOSTIC_TLV_ROUTER_NEIGHBOR = 31
OT_NETWORK_DIAGNOSTIC_TLV_ANSWER = 32
OT_NETWORK_DIAGNOSTIC_TLV_QUERY_ID = 33
OT_NETWORK_DIAGNOSTIC_TLV_MLE_COUNTERS = 34

}

enum otDeviceRole {

OT_DEVICE_ROLE_DISABLED = 0
OT_DEVICE_ROLE_DETACHED = 1
OT_DEVICE_ROLE_CHILD = 2
OT_DEVICE_ROLE_ROUTER = 3
OT_DEVICE_ROLE_LEADER = 4

}

Represents a Thread device role.

Typedefs

typedef uint32_t otNetworkDataIterator

Used to iterate through Network Data information.

typedef struct otBorderRouterConfig otBorderRouterConfig

Represents a Border Router configuration.

typedef struct otLowpanContextInfo

otLowpanContextInfo
Represents 6LoWPAN Context ID information associated with a prefix in Network Data.

typedef struct otExternalRouteConfig

otExternalRouteConfig
Represents an External Route configuration.

typedef enum otRoutePreference

otRoutePreference
Defines valid values for `mPreference` in `otExternalRouteConfig` and `otBorderRouterConfig`.

typedef struct otServerConfig

otServerConfig
Represents a Server configuration.

typedef struct otServiceConfig

otServiceConfig
Represents a Service configuration.

typedef enum otNetDataPublisherEvent

otNetDataPublisherEvent
Represents the events reported from the Publisher callbacks.

typedef void(* otNetDataDnsSrpServicePublisherCallback)(otNetDataPublisherEvent aEvent, void *aContext)

Pointer type defines the callback used to notify when a "DNS/SRP Service" entry is added to or removed from the Thread Network Data.

typedef void(* otNetDataPrefixPublisherCallback)(otNetDataPublisherEvent aEvent, const otIp6Prefix *aPrefix, void *aContext)

Pointer type defines the callback used to notify when a prefix (on-mesh or external route) entry is added to or removed from the Thread Network Data.

typedef uint16_t otNetworkDiagIterator

Used to iterate through Network Diagnostic TLV.

typedef struct otNetworkDiagConnectivity

otNetworkDiagConnectivity
Represents a Network Diagnostic Connectivity value.

typedef struct otNetworkDiagRouteData

otNetworkDiagRouteData
Represents a Network Diagnostic Route data.

typedef struct otNetworkDiagRoute

otNetworkDiagRoute
Represents a Network Diagnostic Route TLV value.

typedef struct otNetworkDiagMacCounters

otNetworkDiagMacCounters
Represents a Network Diagnostic Mac Counters value.

typedef struct otNetworkDiagMleCounters

otNetworkDiagMleCounters
Represents a Network Diagnostics MLE Counters value.

typedef struct otNetworkDiagChildEntry

otNetworkDiagChildEntry
Represents a Network Diagnostic Child Table Entry.

typedef struct otNetworkDiagTlv

otNetworkDiagTlv
Represents a Network Diagnostic TLV.

typedef void(* otReceiveDiagnosticGetCallback)(otError aError, otMessage *aMessage, const otMessageInfo *aMessageInfo, void *aContext)

Pointer is called when Network Diagnostic Get response is received.

typedef struct otLinkModeConfig
Represents an MLE Link Mode configuration.

typedef int16_t otNeighborInfoIterator
Used to iterate through neighbor table.

typedef struct otLeaderData
Represents the Thread Leader Data.

typedef struct otIpCounters
Represents the IP level counters.

typedef struct otMleCounters
Represents the Thread MLE counters.

typedef struct otThreadParentResponseInfo
Represents the MLE Parent Response data.

typedef void(* otDetachGracefullyCallback)(void *aContext)
This callback informs the application that the detaching process has finished.

typedef void(* otThreadParentResponseCallback)(otThreadParentResponseInfo *aInfo, void *aContext)
Pointer is called every time an MLE Parent Response message is received.

typedef struct otThreadDiscoveryRequestInfo
Represents the Thread Discovery Request data.

typedef void(* otThreadDiscoveryRequestCallback)(const otThreadDiscoveryRequestInfo *aInfo, void *aContext)
Pointer is called every time an MLE Discovery Request message is received.

typedef void(* otThreadAnycastLocatorCallback)(void *aContext, otError aError, const otIp6Address *aMeshLocalAddress, uint16_t aRloc16)
Pointer type defines the callback to notify the outcome of a `otThreadLocateAnycastDestination()` request.

Functions

otError otNetDataGet(otInstance *aInstance, bool aStable, uint8_t *aData, uint8_t *aDataLength)
Provide full or stable copy of the Partition's Thread Network Data.

uint8_t otNetDataGetLength(otInstance *aInstance)
Get the current length (number of bytes) of Partition's Thread Network Data.

uint8_t otNetDataGetMaxLength(otInstance *aInstance)
Get the maximum observed length of the Thread Network Data since OT stack initialization or since the last call to `otNetDataResetMaxLength()`.

void otNetDataResetMaxLength(otInstance *aInstance)
Reset the tracked maximum length of the Thread Network Data.

otError otNetDataGetNextOnMeshPrefix(otInstance *aInstance, otNetworkDataIterator *aIterator, otBorderRouterConfig *aConfig)
Get the next On Mesh Prefix in the partition's Network Data.

otError otNetDataGetNextRoute(otInstance *aInstance, otNetworkDataIterator *aIterator, otExternalRouteConfig *aConfig)
Get the next external route in the partition's Network Data.

otError otNetDataGetNextService(otInstance *aInstance, otNetworkDataIterator *aIterator, otServiceConfig *aConfig)
Get the next service in the partition's Network Data.

otError otNetDataGetNextLowpanContextInfo(otInstance *aInstance, otNetworkDataIterator *aIterator, otLowpanContextInfo *aContextInfo)
Get the next 6LoWPAN Context ID info in the partition's Network Data.

void otNetDataGetCommissioningDataset(otInstance *aInstance, otCommissioningDataset *aDataset)
Gets the Commissioning Dataset from the partition's Network Data.

uint8_t otNetDataGetVersion(otInstance *aInstance)
Get the Network Data Version.

uint8_t otNetDataGetStableVersion(otInstance *aInstance)
Get the Stable Network Data Version.

otError otNetDataSteeringDataCheckJoiner(otInstance *aInstance, const otExtAddress *aEui64)
Check if the steering data includes a Joiner.

otError otNetDataSteeringDataCheckJoinerWithDiscerner(otInstance *aInstance, const struct otJoinerDiscerner *aDiscerner)
Check if the steering data includes a Joiner with a given discerner value.

bool otNetDataContainsOmrPrefix(otInstance *aInstance, const otIp6Prefix *aPrefix)
Check whether a given Prefix can act as a valid OMR prefix and also the Leader's Network Data contains this prefix.

void otNetDataPublishDnsSrpServiceAnycast(otInstance *aInstance, uint8_t aSequenceNUmber)
Requests "DNS/SRP Service Anycast Address" to be published in the Thread Network Data.

void otNetDataPublishDnsSrpServiceUnicast(otInstance *aInstance, const otIp6Address *aAddress, uint16_t aPort)
Requests "DNS/SRP Service Unicast Address" to be published in the Thread Network Data.

void otNetDataPublishDnsSrpServiceUnicastMeshLocalEid(otInstance *aInstance, uint16_t aPort)
Requests "DNS/SRP Service Unicast Address" to be published in the Thread Network Data.

bool otNetDataIsDnsSrpServiceAdded(otInstance *aInstance)
Indicates whether or not currently the "DNS/SRP Service" entry is added to the Thread Network Data.

void otNetDataSetDnsSrpServicePublisherCallback(otInstance *aInstance, otNetDataDnsSrpServicePublisherCallback aCallback, void *aContext)
Sets a callback for notifying when a published "DNS/SRP Service" is actually added to or removed from the Thread Network Data.

void otNetDataUnpublishDnsSrpService(otInstance *aInstance)
Unpublishes any previously added DNS/SRP (Anycast or Unicast) Service entry from the Thread Network Data.

otError otNetDataPublishOnMeshPrefix(otInstance *aInstance, const otBorderRouterConfig *aConfig)
Requests an on-mesh prefix to be published in the Thread Network Data.

otError otNetDataPublishExternalRoute(otInstance *aInstance, const otExternalRouteConfig *aConfig)
Requests an external route prefix to be published in the Thread Network Data.

otError otNetDataReplacePublishedExternalRoute(otInstance *aInstance, const otIp6Prefix *aPrefix, const otExternalRouteConfig *aConfig)
Replaces a previously published external route in the Thread Network Data.

bool otNetDataIsPrefixAdded(otInstance *aInstance, const otIp6Prefix *aPrefix)
Indicates whether or not currently a published prefix entry (on-mesh or external route) is added to the Thread Network Data.

void otNetDataSetPrefixPublisherCallback(otInstance *aInstance, otNetDataPrefixPublisherCallback aCallback, void *aContext)
Sets a callback for notifying when a published prefix entry is actually added to or removed from the Thread Network Data.

otError	otNetDataUnpublishPrefix(otInstance *aInstance, const otIp6Prefix *aPrefix) Unpublishes a previously published On-Mesh or External Route Prefix.
otError	otThreadGetNextDiagnosticTlv(const otMessage *aMessage, otNetworkDiagIterator *aIterator, otNetworkDiagTlv *aNetworkDiagTlv) Gets the next Network Diagnostic TLV in the message.
otError	otThreadSendDiagnosticGet(otInstance *aInstance, const otIp6Address *aDestination, const uint8_t aTlvTypes[], uint8_t aCount, otReceiveDiagnosticGetCallback aCallback, void *aCallbackContext) Send a Network Diagnostic Get request.
otError	otThreadSendDiagnosticReset(otInstance *aInstance, const otIp6Address *aDestination, const uint8_t aTlvTypes[], uint8_t aCount) Send a Network Diagnostic Reset request.
const char *	otThreadGetVendorName(otInstance *aInstance) Get the vendor name string.
const char *	otThreadGetVendorModel(otInstance *aInstance) Get the vendor model string.
const char *	otThreadGetVendorSwVersion(otInstance *aInstance) Get the vendor sw version string.
otError	otThreadSetVendorName(otInstance *aInstance, const char *aVendorName) Set the vendor name string.
otError	otThreadSetVendorModel(otInstance *aInstance, const char *aVendorModel) Set the vendor model string.
otError	otThreadSetVendorSwVersion(otInstance *aInstance, const char *aVendorSwVersion) Set the vendor software version string.
otError	otThreadSetEnabled(otInstance *aInstance, bool aEnabled) Starts Thread protocol operation.
uint16_t	otThreadGetVersion(void) Gets the Thread protocol version.
bool	otThreadIsSingleton(otInstance *aInstance) Indicates whether a node is the only router on the network.
otError	otThreadDiscover(otInstance *aInstance, uint32_t aScanChannels, uint16_t aPanId, bool aJoiner, bool aEnableEui64Filtering, otHandleActiveScanResult aCallback, void *aCallbackContext) Starts a Thread Discovery scan.
bool	otThreadIsDiscoverInProgress(otInstance *aInstance) Determines if an MLE Thread Discovery is currently in progress.
otError	otThreadSetJoinerAdvertisement(otInstance *aInstance, uint32_t aOui, const uint8_t *aAdvData, uint8_t aAdvDataLength) Sets the Thread Joiner Advertisement when discovering Thread network.
uint32_t	otThreadGetChildTimeout(otInstance *aInstance) Gets the Thread Child Timeout (in seconds) used when operating in the Child role.
void	otThreadSetChildTimeout(otInstance *aInstance, uint32_t aTimeout) Sets the Thread Child Timeout (in seconds) used when operating in the Child role.
const otExtendedPanId *	otThreadGetExtendedPanId(otInstance *aInstance) Gets the IEEE 802.15.4 Extended PAN ID.

otError	otThreadSetExtendedPanId(otInstance *aInstance, const otExtendedPanId *aExtendedPanId) Sets the IEEE 802.15.4 Extended PAN ID.
otError	otThreadGetLeaderRloc(otInstance *aInstance, otIp6Address *aLeaderRloc) Returns a pointer to the Leader's RLOC.
otLinkModeConfig	otThreadGetLinkMode(otInstance *aInstance) Get the MLE Link Mode configuration.
otError	otThreadSetLinkMode(otInstance *aInstance, otLinkModeConfig aConfig) Set the MLE Link Mode configuration.
void	otThreadGetNetworkKey(otInstance *aInstance, otNetworkKey *aNetworkKey) Get the Thread Network Key.
otNetworkKeyRef	otThreadGetNetworkKeyRef(otInstance *aInstance) Get the `otNetworkKeyRef` for Thread Network Key.
otError	otThreadSetNetworkKey(otInstance *aInstance, const otNetworkKey *aKey) Set the Thread Network Key.
otError	otThreadSetNetworkKeyRef(otInstance *aInstance, otNetworkKeyRef aKeyRef) Set the Thread Network Key as a `otNetworkKeyRef`.
const otIp6Address *	otThreadGetRloc(otInstance *aInstance) Gets the Thread Routing Locator (RLOC) address.
const otIp6Address *	otThreadGetMeshLocalEid(otInstance *aInstance) Gets the Mesh Local EID address.
const otMeshLocalPrefix *	otThreadGetMeshLocalPrefix(otInstance *aInstance) Returns a pointer to the Mesh Local Prefix.
otError	otThreadSetMeshLocalPrefix(otInstance *aInstance, const otMeshLocalPrefix *aMeshLocalPrefix) Sets the Mesh Local Prefix.
const otIp6Address *	otThreadGetLinkLocalIp6Address(otInstance *aInstance) Gets the Thread link-local IPv6 address.
const otIp6Address *	otThreadGetLinkLocalAllThreadNodesMulticastAddress(otInstance *aInstance) Gets the Thread Link-Local All Thread Nodes multicast address.
const otIp6Address *	otThreadGetRealmLocalAllThreadNodesMulticastAddress(otInstance *aInstance) Gets the Thread Realm-Local All Thread Nodes multicast address.
otError	otThreadGetServiceAloc(otInstance *aInstance, uint8_t aServiceId, otIp6Address *aServiceAloc) Retrieves the Service ALOC for given Service ID.
const char *	otThreadGetNetworkName(otInstance *aInstance) Get the Thread Network Name.
otError	otThreadSetNetworkName(otInstance *aInstance, const char *aNetworkName) Set the Thread Network Name.
const char *	otThreadGetDomainName(otInstance *aInstance) Gets the Thread Domain Name.
otError	otThreadSetDomainName(otInstance *aInstance, const char *aDomainName) Sets the Thread Domain Name.
otError	otThreadSetFixedDuaInterfaceIdentifier(otInstance *aInstance, const otIp6InterfaceIdentifier *aId) Sets or clears the Interface Identifier manually specified for the Thread Domain Unicast Address.

const otIp6InterfaceIdentifier *	otThreadGetFixedDualInterfaceIdentifier(otInstance *aInstance) Gets the Interface Identifier manually specified for the Thread Domain Unicast Address.
uint32_t	otThreadGetKeySequenceCounter(otInstance *aInstance) Gets the thrKeySequenceCounter.
void	otThreadSetKeySequenceCounter(otInstance *aInstance, uint32_t aKeySequenceCounter) Sets the thrKeySequenceCounter.
uint32_t	otThreadGetKeySwitchGuardTime(otInstance *aInstance) Gets the thrKeySwitchGuardTime (in hours).
void	otThreadSetKeySwitchGuardTime(otInstance *aInstance, uint32_t aKeySwitchGuardTime) Sets the thrKeySwitchGuardTime (in hours).
otError	otThreadBecomeDetached(otInstance *aInstance) Detach from the Thread network.
otError	otThreadBecomeChild(otInstance *aInstance) Attempt to reattach as a child.
otError	otThreadGetNextNeighborInfo(otInstance *aInstance, otNeighborInfoIterator *aIterator, otNeighborInfo *aInfo) Gets the next neighbor information.
otDeviceRole	otThreadGetDeviceRole(otInstance *aInstance) Get the device role.
const char *	otThreadDeviceRoleToString(otDeviceRole aRole) Convert the device role to human-readable string.
otError	otThreadGetLeaderData(otInstance *aInstance, otLeaderData *aLeaderData) Get the Thread Leader Data.
uint8_t	otThreadGetLeaderRouterId(otInstance *aInstance) Get the Leader's Router ID.
uint8_t	otThreadGetLeaderWeight(otInstance *aInstance) Get the Leader's Weight.
uint32_t	otThreadGetPartitionId(otInstance *aInstance) Get the Partition ID.
uint16_t	otThreadGetRloc16(otInstance *aInstance) Get the RLOC16.
otError	otThreadGetParentInfo(otInstance *aInstance, otRouterInfo *aParentInfo) The function retrieves diagnostic information for a Thread Router as parent.
otError	otThreadGetParentAverageRssi(otInstance *aInstance, int8_t *aParentRssi) The function retrieves the average RSSI for the Thread Parent.
otError	otThreadGetParentLastRssi(otInstance *aInstance, int8_t *aLastRssi) The function retrieves the RSSI of the last packet from the Thread Parent.
otError	otThreadSearchForBetterParent(otInstance *aInstance) Starts the process for child to search for a better parent while staying attached to its current parent.
const otIpCounters *	otThreadGetIp6Counters(otInstance *aInstance) Gets the IPv6 counters.

void	otThreadResetIp6Counters(otInstance *aInstance) Resets the IPv6 counters.
const uint32_t *	otThreadGetTimeInQueueHistogram(otInstance *aInstance, uint16_t *aNumBins, uint32_t *aBinInterval) Gets the time-in-queue histogram for messages in the TX queue.
uint32_t	otThreadGetMaxTimeInQueue(otInstance *aInstance) Gets the maximum time-in-queue for messages in the TX queue.
void	otThreadResetTimeInQueueStat(otInstance *aInstance) Resets the TX queue time-in-queue statistics.
const otMleCounters *	otThreadGetMleCounters(otInstance *aInstance) Gets the Thread MLE counters.
void	otThreadResetMleCounters(otInstance *aInstance) Resets the Thread MLE counters.
void	otThreadRegisterParentResponseCallback(otInstance *aInstance, otThreadParentResponseCallback aCallback, void *aContext) Registers a callback to receive MLE Parent Response data.
void	otThreadSetDiscoveryRequestCallback(otInstance *aInstance, otThreadDiscoveryRequestCallback aCallback, void *aContext) Sets a callback to receive MLE Discovery Request data.
otError	otThreadLocateAnycastDestination(otInstance *aInstance, const otIp6Address *aAnycastAddress, otThreadAnycastLocatorCallback aCallback, void *aContext) Requests the closest destination of a given anycast address to be located.
bool	otThreadIsAnycastLocateInProgress(otInstance *aInstance) Indicates whether an anycast locate request is currently in progress.
void	otThreadSendAddressNotification(otInstance *aInstance, otIp6Address *aDestination, otIp6Address *aTarget, otIp6InterfaceIdentifier *aMlIid) Sends a Proactive Address Notification (ADDR_NTF.ntf) message.
otError	otThreadSendProactiveBackboneNotification(otInstance *aInstance, otIp6Address *aTarget, otIp6InterfaceIdentifier *aMlIid, uint32_t aTimeSinceLastTransaction) Sends a Proactive Backbone Notification (PRO_BB.ntf) message on the Backbone link.
otError	otThreadDetachGracefully(otInstance *aInstance, otDetachGracefullyCallback aCallback, void *aContext) Notifies other nodes in the network (if any) and then stops Thread protocol operation.
void	otConvertDurationInSecondsToString(uint32_t aDuration, char *aBuffer, uint16_t aSize) Converts an `uint32_t` duration (in seconds) to a human-readable string.

Macros

#define	OT_NETWORK_DATA_ITERATOR_INIT 0 Value to initialize `otNetworkDataIterator`.
#define	OT_SERVICE_DATA_MAX_SIZE 252 Max size of Service Data in bytes.
#define	OT_SERVER_DATA_MAX_SIZE 248 Max size of Server Data in bytes. Theoretical limit, practically much lower.
#define	OT_NETWORK_DIAGNOSTIC_TYPELIST_MAX_ENTRIES 19 Maximum Number of Network Diagnostic TLV Types to Request or Reset.

#define OT_NETWORK_DIAGNOSTIC_CHILD_TABLE_ENTRY_SIZE 3
Size of Network Diagnostic Child Table entry.

#define OT_NETWORK_DIAGNOSTIC_ITERATOR_INIT 0
Initializer for otNetworkDiagIterator.

#define OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_NAME_TLV_LENGTH 32
Max length of Vendor Name TLV.

#define OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_MODEL_TLV_LENGTH 32
Max length of Vendor Model TLV.

#define OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_SW_VERSION_TLV_LENGTH 16
Max length of Vendor SW Version TLV.

#define OT_NETWORK_DIAGNOSTIC_MAX_THREAD_STACK_VERSION_TLV_LENGTH 64
Max length of Thread Stack Version TLV.

#define OT_NETWORK_BASE_TLV_MAX_LENGTH 254
Maximum value length of Thread Base TLV.

#define OT_NETWORK_MAX_ROUTER_ID 62
Maximum Router ID.

#define OT_NEIGHBOR_INFO_ITERATOR_INIT 0
Initializer for otNeighborInfoIterator.

#define OT_JOINER_ADVDATA_MAX_LENGTH 64
Maximum AdvData Length of Joiner Advertisement.

#define OT_DURATION_STRING_SIZE 21
Recommended size for string representation of `uint32_t` duration in seconds.

Enumeration Documentation

otRoutePreference

> otRoutePreference

Defines valid values for `mPreference` in `otExternalRouteConfig` and `otBorderRouterConfig`.

Enumerator	
OT_ROUTE_PREFERENCE_LOW	Low route preference.
OT_ROUTE_PREFERENCE_MED	Medium route preference.
OT_ROUTE_PREFERENCE_HIGH	High route preference.

Definition at line `105` of file `include/openthread/netdata.h`

otNetDataPublisherEvent

> otNetDataPublisherEvent

Represents the events reported from the Publisher callbacks.

Enumerator	
OT_NETDATA_PUBLISHER_EVENT_ENTRY_ADDED	Published entry is added to the Thread Network Data.
OT_NETDATA_PUBLISHER_EVENT_ENTRY_REMOVED	Published entry is removed from the Thread Network Data.

Definition at line 61 of file include/openthread/netdata_publisher.h

@5

> @5

Enumerator	
OT_NETWORK_DIAGNOSTIC_TLV_EXT_ADDRESS	MAC Extended Address TLV.
OT_NETWORK_DIAGNOSTIC_TLV_SHORT_ADDRESS	Address16 TLV.
OT_NETWORK_DIAGNOSTIC_TLV_MODE	Mode TLV.
OT_NETWORK_DIAGNOSTIC_TLV_TIMEOUT	Timeout TLV (the maximum polling time period for SEDs)
OT_NETWORK_DIAGNOSTIC_TLV_CONNECTIVITY	Connectivity TLV.
OT_NETWORK_DIAGNOSTIC_TLV_ROUTE	Route64 TLV.
OT_NETWORK_DIAGNOSTIC_TLV_LEADER_DATA	Leader Data TLV.
OT_NETWORK_DIAGNOSTIC_TLV_NETWORK_DATA	Network Data TLV.
OT_NETWORK_DIAGNOSTIC_TLV_IP6_ADDR_LIST	IPv6 Address List TLV.
OT_NETWORK_DIAGNOSTIC_TLV_MAC_COUNTERS	MAC Counters TLV.
OT_NETWORK_DIAGNOSTIC_TLV_BATTERY_LEVEL	Battery Level TLV.
OT_NETWORK_DIAGNOSTIC_TLV_SUPPLY_VOLTAGE	Supply Voltage TLV.
OT_NETWORK_DIAGNOSTIC_TLV_CHILD_TABLE	Child Table TLV.
OT_NETWORK_DIAGNOSTIC_TLV_CHANNEL_PAGES	Channel Pages TLV.
OT_NETWORK_DIAGNOSTIC_TLV_TYPE_LIST	Type List TLV.
OT_NETWORK_DIAGNOSTIC_TLV_MAX_CHILD_TIMEOUT	Max Child Timeout TLV.
OT_NETWORK_DIAGNOSTIC_TLV_VERSION	Version TLV.
OT_NETWORK_DIAGNOSTIC_TLV_VENDOR_NAME	Vendor Name TLV.
OT_NETWORK_DIAGNOSTIC_TLV_VENDOR_MODEL	Vendor Model TLV.
OT_NETWORK_DIAGNOSTIC_TLV_VENDOR_SW_VERSION	Vendor SW Version TLV.
OT_NETWORK_DIAGNOSTIC_TLV_THREAD_STACK_VERSION	Thread Stack Version TLV.
OT_NETWORK_DIAGNOSTIC_TLV_CHILD	Child TLV.
OT_NETWORK_DIAGNOSTIC_TLV_CHILD_IP6_ADDR_LIST	Child IPv6 Address List TLV.
OT_NETWORK_DIAGNOSTIC_TLV_ROUTER_NEIGHBOR	Router Neighbor TLV.
OT_NETWORK_DIAGNOSTIC_TLV_ANSWER	Answer TLV.
OT_NETWORK_DIAGNOSTIC_TLV_QUERY_ID	Query ID TLV.
OT_NETWORK_DIAGNOSTIC_TLV_MLE_COUNTERS	MLE Counters TLV.

Definition at line 67 of file include/openthread/netdiag.h

otDeviceRole

> otDeviceRole

Represents a Thread device role.

Enumerator	
OT_DEVICE_ROLE_DISABLED	The Thread stack is disabled.

OT_DEVICE_ROLE_DETACHED	Not currently participating in a Thread network/partition.
OT_DEVICE_ROLE_CHILD	The Thread Child role.
OT_DEVICE_ROLE_ROUTER	The Thread Router role.
OT_DEVICE_ROLE_LEADER	The Thread Leader role.

Definition at line `67` of file `include/openthread/thread.h`

Typedef Documentation

otNetworkDataIterator

```
typedef uint32_t otNetworkDataIterator
```

Used to iterate through Network Data information.

Definition at line `54` of file `include/openthread/netdata.h`

otBorderRouterConfig

```
typedef struct otBorderRouterConfig otBorderRouterConfig
```

Represents a Border Router configuration.

Definition at line `73` of file `include/openthread/netdata.h`

otLowpanContextInfo

```
typedef struct otLowpanContextInfo otLowpanContextInfo
```

Represents 6LoWPAN Context ID information associated with a prefix in Network Data.

Definition at line `84` of file `include/openthread/netdata.h`

otExternalRouteConfig

```
typedef struct otExternalRouteConfig otExternalRouteConfig
```

Represents an External Route configuration.

Definition at line `99` of file `include/openthread/netdata.h`

otRoutePreference

```
typedef enum otRoutePreference otRoutePreference
```

Defines valid values for `mPreference` in `otExternalRouteConfig` and `otBorderRouterConfig`.

Definition at line `110` of file `include/openthread/netdata.h`

otServerConfig

```
typedef struct otServerConfig otServerConfig
```

Represents a Server configuration.

Definition at line `125` of file `include/openthread/netdata.h`

otServiceConfig

```
typedef struct otServiceConfig otServiceConfig
```

Represents a Service configuration.

Definition at line `138` of file `include/openthread/netdata.h`

otNetDataPublisherEvent

```
typedef enum otNetDataPublisherEvent otNetDataPublisherEvent
```

Represents the events reported from the Publisher callbacks.

Definition at line `65` of file `include/openthread/netdata_publisher.h`

otNetDataDnsSrpServicePublisherCallback

```
typedef void(* otNetDataDnsSrpServicePublisherCallback) (otNetDataPublisherEvent aEvent, void *aContext) )
(otNetDataPublisherEvent aEvent, void *aContext)
```

Pointer type defines the callback used to notify when a "DNS/SRP Service" entry is added to or removed from the Thread Network Data.

Parameters

[in]	aEvent	Indicates the event (whether the entry was added or removed).
[in]	aContext	A pointer to application-specific context.

On remove the callback is invoked independent of whether the entry is removed by `Publisher` (e.g., when there are too many similar entries already present in the Network Data) or through an explicit call to unpublish the entry (i.e., a call to `otNetDataUnpublishDnsSrpService()`).

Definition at line `79` of file `include/openthread/netdata_publisher.h`

otNetDataPrefixPublisherCallback

```
typedef void(* otNetDataPrefixPublisherCallback) (otNetDataPublisherEvent aEvent, const otIp6Prefix *aPrefix, void
*aContext) )(otNetDataPublisherEvent aEvent, const otIp6Prefix *aPrefix, void *aContext)
```

Pointer type defines the callback used to notify when a prefix (on-mesh or external route) entry is added to or removed from the Thread Network Data.

Parameters

[in]	aEvent	Indicates the event (whether the entry was added or removed).
[in]	aPrefix	A pointer to the prefix entry.
[in]	aContext	A pointer to application-specific context.

On remove the callback is invoked independent of whether the entry is removed by `Publisher` (e.g., when there are too many similar entries already present in the Network Data) or through an explicit call to unpublish the entry.

Definition at line `93` of file `include/openthread/netdata_publisher.h`

otNetworkDiagIterator

```
typedef uint16_t otNetworkDiagIterator
```

Used to iterate through Network Diagnostic TLV.

Definition at line `104` of file `include/openthread/netdiag.h`

otNetworkDiagConnectivity

```
typedef struct otNetworkDiagConnectivity otNetworkDiagConnectivity
```

Represents a Network Diagnostic Connectivity value.

Definition at line `156` of file `include/openthread/netdiag.h`

otNetworkDiagRouteData

```
typedef struct otNetworkDiagRouteData otNetworkDiagRouteData
```

Represents a Network Diagnostic Route data.

Definition at line `168` of file `include/openthread/netdiag.h`

otNetworkDiagRoute

```
typedef struct otNetworkDiagRoute otNetworkDiagRoute
```

Represents a Network Diagnostic Route TLV value.

Definition at line `190` of file `include/openthread/netdiag.h`

otNetworkDiagMacCounters

```
typedef struct otNetworkDiagMacCounters otNetworkDiagMacCounters
```

Represents a Network Diagnostic Mac Counters value.

See RFC 2863 for definitions of member fields.

Definition at line `209` of file `include/openthread/netdiag.h`

otNetworkDiagMleCounters

```
typedef struct otNetworkDiagMleCounters otNetworkDiagMleCounters
```

Represents a Network Diagnostics MLE Counters value.

Definition at line `232` of file `include/openthread/netdiag.h`

otNetworkDiagChildEntry

```
typedef struct otNetworkDiagChildEntry otNetworkDiagChildEntry
```

Represents a Network Diagnostic Child Table Entry.

Definition at line `262` of file `include/openthread/netdiag.h`

otNetworkDiagTlv

```
typedef struct otNetworkDiagTlv otNetworkDiagTlv
```

Represents a Network Diagnostic TLV.

Definition at line `316` of file `include/openthread/netdiag.h`

otReceiveDiagnosticGetCallback

```
typedef void(* otReceiveDiagnosticGetCallback) (otError aError, otMessage *aMessage, const otMessageInfo
*aMessageInfo, void *aContext) )(otError aError, otMessage *aMessage, const otMessageInfo *aMessageInfo, void
*aContext)
```

Pointer is called when Network Diagnostic Get response is received.

Parameters

[in]	aError	The error when failed to get the response.
[in]	aMessage	A pointer to the message buffer containing the received Network Diagnostic Get response payload. Available only when `aError` is `OT_ERROR_NONE`.
[in]	aMessageInfo	A pointer to the message info for `aMessage`. Available only when `aError` is `OT_ERROR_NONE`.
[in]	aContext	A pointer to application-specific context.

Definition at line `350` of file `include/openthread/netdiag.h`

otLinkModeConfig

```
typedef struct otLinkModeConfig otLinkModeConfig
```

Represents an MLE Link Mode configuration.

Definition at line `84` of file `include/openthread/thread.h`

otNeighborInfoIterator

```
typedef int16_t otNeighborInfoIterator
```

Used to iterate through neighbor table.

Definition at line 113 of file include/openthread/thread.h

otLeaderData

```
typedef struct otLeaderData otLeaderData
```

Represents the Thread Leader Data.

Definition at line 126 of file include/openthread/thread.h

otIpCounters

```
typedef struct otIpCounters otIpCounters
```

Represents the IP level counters.

Definition at line 164 of file include/openthread/thread.h

otMleCounters

```
typedef struct otMleCounters otMleCounters
```

Represents the Thread MLE counters.

Definition at line 203 of file include/openthread/thread.h

otThreadParentResponseInfo

```
typedef struct otThreadParentResponseInfo otThreadParentResponseInfo
```

Represents the MLE Parent Response data.

Definition at line 219 of file include/openthread/thread.h

otDetachGracefullyCallback

```
typedef void(* otDetachGracefullyCallback) (void *aContext) )(void *aContext)
```

This callback informs the application that the detaching process has finished.

Parameters

[in]	aContext	A pointer to application-specific context.

Definition at line 227 of file include/openthread/thread.h

otThreadParentResponseCallback

```
typedef void(* otThreadParentResponseCallback) (otThreadParentResponseInfo *aInfo, void *aContext) )
(otThreadParentResponseInfo *aInfo, void *aContext)
```

Pointer is called every time an MLE Parent Response message is received.

Parameters

[in]	aInfo	A pointer to a location on stack holding the stats data.
[in]	aContext	A pointer to callback client-specific context.

This is used in otThreadRegisterParentResponseCallback() .

Definition at line 989 of file include/openthread/thread.h

otThreadDiscoveryRequestInfo

```
typedef struct otThreadDiscoveryRequestInfo otThreadDiscoveryRequestInfo
```

Represents the Thread Discovery Request data.

Definition at line 1014 of file include/openthread/thread.h

otThreadDiscoveryRequestCallback

```
typedef void(* otThreadDiscoveryRequestCallback) (const otThreadDiscoveryRequestInfo *aInfo, void *aContext) )(const
otThreadDiscoveryRequestInfo *aInfo, void *aContext)
```

Pointer is called every time an MLE Discovery Request message is received.

Parameters

[in]	aInfo	A pointer to the Discovery Request info data.
[in]	aContext	A pointer to callback application-specific context.

Definition at line 1023 of file include/openthread/thread.h

otThreadAnycastLocatorCallback

```
typedef void(* otThreadAnycastLocatorCallback) (void *aContext, otError aError, const otIp6Address
*aMeshLocalAddress, uint16_t aRloc16) )(void *aContext, otError aError, const otIp6Address *aMeshLocalAddress, uint16_t
aRloc16)
```

Pointer type defines the callback to notify the outcome of a otThreadLocateAnycastDestination() request.

Parameters

[in]	aContext	A pointer to an arbitrary context (provided when callback is registered).
[in]	aError	The error when handling the request. OT_ERROR_NONE indicates success. OT_ERROR_RESPONSE_TIMEOUT indicates a destination could not be found. OT_ERROR_ABORT indicates the request was aborted.

[in]	aMeshLocalAddress	A pointer to the mesh-local EID of the closest destination of the anycast address when `aError` is OT_ERROR_NONE, NULL otherwise.
[in]	aRloc16	The RLOC16 of the destination if found, otherwise invalid RLOC16 (0xfffe).

Definition at line `1050` of file `include/openthread/thread.h`

Function Documentation

otNetDataGet

```
otError otNetDataGet (otInstance *aInstance, bool aStable, uint8_t *aData, uint8_t *aDataLength)
```

Provide full or stable copy of the Partition's Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aStable	TRUE when copying the stable version, FALSE when copying the full version.
[out]	aData	A pointer to the data buffer.
[inout]	aDataLength	On entry, size of the data buffer pointed to by `aData`. On exit, number of copied bytes.

Definition at line `153` of file `include/openthread/netdata.h`

otNetDataGetLength

```
uint8_t otNetDataGetLength (otInstance *aInstance)
```

Get the current length (number of bytes) of Partition's Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The length of the Network Data.

Definition at line `163` of file `include/openthread/netdata.h`

otNetDataGetMaxLength

```
uint8_t otNetDataGetMaxLength (otInstance *aInstance)
```

Get the maximum observed length of the Thread Network Data since OT stack initialization or since the last call to `otNetDataResetMaxLength()`.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The maximum length of the Network Data (high water mark for Network Data length).

Definition at line 174 of file include/openthread/netdata.h

otNetDataResetMaxLength

```
void otNetDataResetMaxLength (otInstance *aInstance)
```

Reset the tracked maximum length of the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

See Also

- otNetDataGetMaxLength

Definition at line 184 of file include/openthread/netdata.h

otNetDataGetNextOnMeshPrefix

```
otError otNetDataGetNextOnMeshPrefix (otInstance *aInstance, otNetworkDataIterator *aIterator, otBorderRouterConfig *aConfig)
```

Get the next On Mesh Prefix in the partition's Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the Network Data iterator context. To get the first on-mesh entry it should be set to OT_NETWORK_DATA_ITERATOR_INIT.
[out]	aConfig	A pointer to where the On Mesh Prefix information will be placed.

Definition at line 198 of file include/openthread/netdata.h

otNetDataGetNextRoute

```
otError otNetDataGetNextRoute (otInstance *aInstance, otNetworkDataIterator *aIterator, otExternalRouteConfig *aConfig)
```

Get the next external route in the partition's Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the Network Data iterator context. To get the first external route entry it should be set to OT_NETWORK_DATA_ITERATOR_INIT.
[out]	aConfig	A pointer to where the External Route information will be placed.

Definition at line 214 of file include/openthread/netdata.h

otNetDataGetNextService

```
otError otNetDataGetNextService (otInstance *aInstance, otNetworkDataIterator *aIterator, otServiceConfig *aConfig)
```

Get the next service in the partition's Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the Network Data iterator context. To get the first service entry it should be set to OT_NETWORK_DATA_ITERATOR_INIT.
[out]	aConfig	A pointer to where the service information will be placed.

Definition at line 228 of file include/openthread/netdata.h

otNetDataGetNextLowpanContextInfo

```
otError otNetDataGetNextLowpanContextInfo (otInstance *aInstance, otNetworkDataIterator *aIterator,
otLowpanContextInfo *aContextInfo)
```

Get the next 6LoWPAN Context ID info in the partition's Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the Network Data iterator. To get the first service entry it should be set to OT_NETWORK_DATA_ITERATOR_INIT.
[out]	aContextInfo	A pointer to where the retrieved 6LoWPAN Context ID information will be placed.

Definition at line 242 of file include/openthread/netdata.h

otNetDataGetCommissioningDataset

```
void otNetDataGetCommissioningDataset (otInstance *aInstance, otCommissioningDataset *aDataset)
```

Gets the Commissioning Dataset from the partition's Network Data.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[out]	aDataset	A pointer to a otCommissioningDataset to populate.

Definition at line 253 of file include/openthread/netdata.h

otNetDataGetVersion

```
uint8_t otNetDataGetVersion (otInstance *aInstance)
```

Get the Network Data Version.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Network Data Version.

Definition at line 263 of file include/openthread/netdata.h

otNetDataGetStableVersion

```
uint8_t otNetDataGetStableVersion (otInstance *aInstance)
```

Get the Stable Network Data Version.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Stable Network Data Version.

Definition at line 273 of file include/openthread/netdata.h

otNetDataSteeringDataCheckJoiner

```
otError otNetDataSteeringDataCheckJoiner (otInstance *aInstance, const otExtAddress *aEui64)
```

Check if the steering data includes a Joiner.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEui64	A pointer to the Joiner's IEEE EUI-64.

Definition at line 286 of file include/openthread/netdata.h

otNetDataSteeringDataCheckJoinerWithDiscerner

```
otError otNetDataSteeringDataCheckJoinerWithDiscerner (otInstance *aInstance, const struct otJoinerDiscerner
*aDiscerner)
```

Check if the steering data includes a Joiner with a given discerner value.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDiscerner	A pointer to the Joiner Discerner.

Definition at line 302 of file include/openthread/netdata.h

otNetDataContainsOmrPrefix

```
bool otNetDataContainsOmrPrefix (otInstance *aInstance, const otIp6Prefix *aPrefix)
```

Check whether a given Prefix can act as a valid OMR prefix and also the Leader's Network Data contains this prefix.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPrefix	A pointer to the IPv6 prefix.

Returns

- Whether `aPrefix` is a valid OMR prefix and Leader's Network Data contains the OMR prefix `aPrefix`.

Note

- This API is only available when `OPENTHREAD_CONFIG_BORDER_ROUTING_ENABLE` is used.

Definition at line `316` of file `include/openthread/netdata.h`

otNetDataPublishDnsSrpServiceAnycast

```
void otNetDataPublishDnsSrpServiceAnycast (otInstance *aInstance, uint8_t aSequenceNUmber)
```

Requests "DNS/SRP Service Anycast Address" to be published in the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSequenceNUmber	The sequence number of DNS/SRP Anycast Service.

Requires the feature `OPENTHREAD_CONFIG_TMF_NETDATA_SERVICE_ENABLE` to be enabled.

A call to this function will remove and replace any previous "DNS/SRP Service" entry that was being published (from earlier call to any of `otNetDataPublishDnsSrpService{Type}()` functions).

Definition at line `109` of file `include/openthread/netdata_publisher.h`

otNetDataPublishDnsSrpServiceUnicast

```
void otNetDataPublishDnsSrpServiceUnicast (otInstance *aInstance, const otIp6Address *aAddress, uint16_t aPort)
```

Requests "DNS/SRP Service Unicast Address" to be published in the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aAddress	The DNS/SRP server address to publish (MUST NOT be NULL).
[in]	aPort	The SRP server port number to publish.

Requires the feature `OPENTHREAD_CONFIG_TMF_NETDATA_SERVICE_ENABLE` to be enabled.

A call to this function will remove and replace any previous "DNS/SRP Service" entry that was being published (from earlier call to any of `otNetDataPublishDnsSrpService{Type}()` functions).

Publishes the "DNS/SRP Service Unicast Address" by including the address and port info in the Service TLV data.

Definition at line `127` of file `include/openthread/netdata_publisher.h`

otNetDataPublishDnsSrpServiceUnicastMeshLocalEid

```
void otNetDataPublishDnsSrpServiceUnicastMeshLocalEid (otInstance *aInstance, uint16_t aPort)
```

Requests "DNS/SRP Service Unicast Address" to be published in the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPort	The SRP server port number to publish.

Requires the feature `OPENTHREAD_CONFIG_TMF_NETDATA_SERVICE_ENABLE` to be enabled.

A call to this function will remove and replace any previous "DNS/SRP Service" entry that was being published (from earlier call to any of `otNetDataPublishDnsSrpService{Type}()` functions).

Unlike `otNetDataPublishDnsSrpServiceUnicast()` which requires the published address to be given and includes the info in the Service TLV data, this function uses the device's mesh-local EID and includes the info in the Server TLV data.

Definition at line `145` of file `include/openthread/netdata_publisher.h`

otNetDataIsDnsSrpServiceAdded

```
bool otNetDataIsDnsSrpServiceAdded (otInstance *aInstance)
```

Indicates whether or not currently the "DNS/SRP Service" entry is added to the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Requires the feature `OPENTHREAD_CONFIG_TMF_NETDATA_SERVICE_ENABLE` to be enabled.

Definition at line `158` of file `include/openthread/netdata_publisher.h`

otNetDataSetDnsSrpServicePublisherCallback

```
void otNetDataSetDnsSrpServicePublisherCallback (otInstance *aInstance, otNetDataDnsSrpServicePublisherCallback aCallback, void *aContext)
```

Sets a callback for notifying when a published "DNS/SRP Service" is actually added to or removed from the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	The callback function pointer (can be NULL if not needed).
[in]	aContext	A pointer to application-specific context (used when `aCallback` is invoked).

A subsequent call to this function replaces any previously set callback function.

Requires the feature `OPENTHREAD_CONFIG_TMF_NETDATA_SERVICE_ENABLE` to be enabled.

Definition at line `173` of file `include/openthread/netdata_publisher.h`

otNetDataUnpublishDnsSrpService

```
void otNetDataUnpublishDnsSrpService (otInstance *aInstance)
```

Unpublishes any previously added DNS/SRP (Anycast or Unicast) Service entry from the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

OPENTHREAD_CONFIG_TMF_NETDATA_SERVICE_ENABLE must be enabled.

Definition at line 186 of file include/openthread/netdata_publisher.h

otNetDataPublishOnMeshPrefix

> otError otNetDataPublishOnMeshPrefix (otInstance *aInstance, const otBorderRouterConfig *aConfig)

Requests an on-mesh prefix to be published in the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aConfig	The on-mesh prefix config to publish (MUST NOT be NULL).

Requires the feature OPENTHREAD_CONFIG_BORDER_ROUTER_ENABLE to be enabled.

Only stable entries can be published (i.e., aConfig.mStable MUST be TRUE).

A subsequent call to this method will replace a previous request for the same prefix. In particular, if the new call only changes the flags (e.g., preference level) and the prefix is already added in the Network Data, the change to flags is immediately reflected in the Network Data. This ensures that existing entries in the Network Data are not abruptly removed. Note that a change in the preference level can potentially later cause the entry to be removed from the Network Data after determining there are other nodes that are publishing the same prefix with the same or higher preference.

Definition at line 213 of file include/openthread/netdata_publisher.h

otNetDataPublishExternalRoute

> otError otNetDataPublishExternalRoute (otInstance *aInstance, const otExternalRouteConfig *aConfig)

Requests an external route prefix to be published in the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aConfig	The external route config to publish (MUST NOT be NULL).

Requires the feature OPENTHREAD_CONFIG_BORDER_ROUTER_ENABLE to be enabled.

Only stable entries can be published (i.e., aConfig.mStable MUST be TRUE).

A subsequent call to this method will replace a previous request for the same prefix. In particular, if the new call only changes the flags (e.g., preference level) and the prefix is already added in the Network Data, the change to flags is immediately reflected in the Network Data. This ensures that existing entries in the Network Data are not abruptly removed. Note that a change in the preference level can potentially later cause the entry to be removed from the Network Data after determining there are other nodes that are publishing the same prefix with the same or higher preference.

Definition at line 238 of file include/openthread/netdata_publisher.h

otNetDataReplacePublishedExternalRoute

> otError otNetDataReplacePublishedExternalRoute (otInstance *aInstance, const otIp6Prefix *aPrefix, const otExternalRouteConfig *aConfig)

Replaces a previously published external route in the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPrefix	The previously published external route prefix to replace.
[in]	aConfig	The external route config to publish.

Requires the feature `OPENTHREAD_CONFIG_BORDER_ROUTER_ENABLE` to be enabled.

If there is no previously published external route matching `aPrefix`, this function behaves similarly to `otNetDataPublishExternalRoute()`, i.e., it will start the process of publishing **aConfig** as an external route in the Thread Network Data.

If there is a previously published route entry matching `aPrefix`, it will be replaced with the new prefix from `aConfig`.

- If the `aPrefix` was already added in the Network Data, the change to the new prefix in `aConfig` is immediately reflected in the Network Data. This ensures that route entries in the Network Data are not abruptly removed and the transition from aPrefix to the new prefix is smooth.
- If the old published `aPrefix` was not added in the Network Data, it will be replaced with the new `aConfig` prefix but it will not be immediately added. Instead, it will start the process of publishing it in the Network Data (monitoring the Network Data to determine when/if to add the prefix, depending on the number of similar prefixes present in the Network Data).

Definition at line `272` of file `include/openthread/netdata_publisher.h`

otNetDataIsPrefixAdded

```
bool otNetDataIsPrefixAdded (otInstance *aInstance, const otIp6Prefix *aPrefix)
```

Indicates whether or not currently a published prefix entry (on-mesh or external route) is added to the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPrefix	A pointer to the prefix (MUST NOT be NULL).

Requires the feature `OPENTHREAD_CONFIG_BORDER_ROUTER_ENABLE` to be enabled.

Definition at line `289` of file `include/openthread/netdata_publisher.h`

otNetDataSetPrefixPublisherCallback

```
void otNetDataSetPrefixPublisherCallback (otInstance *aInstance, otNetDataPrefixPublisherCallback aCallback, void *aContext)
```

Sets a callback for notifying when a published prefix entry is actually added to or removed from the Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	The callback function pointer (can be NULL if not needed).
[in]	aContext	A pointer to application-specific context (used when `aCallback` is invoked).

A subsequent call to this function replaces any previously set callback function.

Requires the feature `OPENTHREAD_CONFIG_BORDER_ROUTER_ENABLE` to be enabled.

Definition at line `304` of file `include/openthread/netdata_publisher.h`

otNetDataUnpublishPrefix

> otError otNetDataUnpublishPrefix (otInstance *aInstance, const otIp6Prefix *aPrefix)

Unpublishes a previously published On-Mesh or External Route Prefix.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPrefix	The prefix to unpublish (MUST NOT be NULL).

`OPENTHREAD_CONFIG_BORDER_ROUTER_ENABLE` must be enabled.

Definition at line `320` of file `include/openthread/netdata_publisher.h`

otThreadGetNextDiagnosticTlv

> otError otThreadGetNextDiagnosticTlv (const otMessage *aMessage, otNetworkDiagIterator *aIterator, otNetworkDiagTlv *aNetworkDiagTlv)

Gets the next Network Diagnostic TLV in the message.

Parameters

[in]	aMessage	A pointer to a message.
[inout]	aIterator	A pointer to the Network Diagnostic iterator context. To get the first Network Diagnostic TLV it should be set to OT_NETWORK_DIAGNOSTIC_ITERATOR_INIT.
[out]	aNetworkDiagTlv	A pointer to where the Network Diagnostic TLV information will be placed.

Requires `OPENTHREAD_CONFIG_TMF_NETDIAG_CLIENT_ENABLE`.

@Note A subsequent call to this function is allowed only when current return value is OT_ERROR_NONE.

Definition at line `335` of file `include/openthread/netdiag.h`

otThreadSendDiagnosticGet

> otError otThreadSendDiagnosticGet (otInstance *aInstance, const otIp6Address *aDestination, const uint8_t aTlvTypes[], uint8_t aCount, otReceiveDiagnosticGetCallback aCallback, void *aCallbackContext)

Send a Network Diagnostic Get request.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDestination	A pointer to destination address.
[in]	aTlvTypes	An array of Network Diagnostic TLV types.
[in]	aCount	Number of types in aTlvTypes.
[in]	aCallback	A pointer to a function that is called when Network Diagnostic Get response is received or NULL to disable the callback.
[in]	aCallbackContext	A pointer to application-specific context.

Requires `OPENTHREAD_CONFIG_TMF_NETDIAG_CLIENT_ENABLE`.

Definition at line 372 of file include/openthread/netdiag.h

otThreadSendDiagnosticReset

```
otError otThreadSendDiagnosticReset (otInstance *aInstance, const otIp6Address *aDestination, const uint8_t aTlvTypes[], uint8_t aCount)
```

Send a Network Diagnostic Reset request.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDestination	A pointer to destination address.
[in]	aTlvTypes	An array of Network Diagnostic TLV types. Currently only Type 9 is allowed.
[in]	aCount	Number of types in aTlvTypes

Requires OPENTHREAD_CONFIG_TMF_NETDIAG_CLIENT_ENABLE .

Definition at line 393 of file include/openthread/netdiag.h

otThreadGetVendorName

```
const char * otThreadGetVendorName (otInstance *aInstance)
```

Get the vendor name string.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The vendor name string.

Definition at line 406 of file include/openthread/netdiag.h

otThreadGetVendorModel

```
const char * otThreadGetVendorModel (otInstance *aInstance)
```

Get the vendor model string.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The vendor model string.

Definition at line 416 of file include/openthread/netdiag.h

otThreadGetVendorSwVersion

```
const char * otThreadGetVendorSwVersion (otInstance *aInstance)
```

Get the vendor sw version string.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The vendor sw version string.

Definition at line `426` of file `include/openthread/netdiag.h`

otThreadSetVendorName

> otError otThreadSetVendorName (otInstance *aInstance, const char *aVendorName)

Set the vendor name string.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aVendorName	The vendor name string.

Requires `OPENTHREAD_CONFIG_NET_DIAG_VENDOR_INFO_SET_API_ENABLE`.

`aVendorName` should be UTF8 with max length of 32 chars (`MAX_VENDOR_NAME_TLV_LENGTH`). Maximum length does not include the null `\0` character.

Definition at line `443` of file `include/openthread/netdiag.h`

otThreadSetVendorModel

> otError otThreadSetVendorModel (otInstance *aInstance, const char *aVendorModel)

Set the vendor model string.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aVendorModel	The vendor model string.

Requires `OPENTHREAD_CONFIG_NET_DIAG_VENDOR_INFO_SET_API_ENABLE`.

`aVendorModel` should be UTF8 with max length of 32 chars (`MAX_VENDOR_MODEL_TLV_LENGTH`). Maximum length does not include the null `\0` character.

Definition at line `460` of file `include/openthread/netdiag.h`

otThreadSetVendorSwVersion

> otError otThreadSetVendorSwVersion (otInstance *aInstance, const char *aVendorSwVersion)

Set the vendor software version string.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[in]	aVendorSwVersion	The vendor software version string.

Requires `OPENTHREAD_CONFIG_NET_DIAG_VENDOR_INFO_SET_API_ENABLE`.

`aVendorSwVersion` should be UTF8 with max length of 16 chars(`MAX_VENDOR_SW_VERSION_TLV_LENGTH`). Maximum length does not include the null `\0` character.

Definition at line `477` of file `include/openthread/netdiag.h`

otThreadSetEnabled

```
otError otThreadSetEnabled (otInstance *aInstance, bool aEnabled)
```

Starts Thread protocol operation.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	TRUE if Thread is enabled, FALSE otherwise.

The interface must be up when calling this function.

Calling this function with `aEnabled` set to FALSE stops any ongoing processes of detaching started by otThreadDetachGracefully(). Its callback will be called.

Definition at line `244` of file `include/openthread/thread.h`

otThreadGetVersion

```
uint16_t otThreadGetVersion (void)
```

Gets the Thread protocol version.

Parameters

N/A		

Returns

- the Thread protocol version.

Definition at line `252` of file `include/openthread/thread.h`

otThreadIsSingleton

```
bool otThreadIsSingleton (otInstance *aInstance)
```

Indicates whether a node is the only router on the network.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `263` of file `include/openthread/thread.h`

otThreadDiscover

> otError otThreadDiscover (otInstance *aInstance, uint32_t aScanChannels, uint16_t aPanId, bool aJoiner, bool aEnableEui64Filtering, otHandleActiveScanResult aCallback, void *aCallbackContext)

Starts a Thread Discovery scan.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aScanChannels	A bit vector indicating which channels to scan (e.g. OT_CHANNEL_11_MASK).
[in]	aPanId	The PAN ID filter (set to Broadcast PAN to disable filter).
[in]	aJoiner	Value of the Joiner Flag in the Discovery Request TLV.
[in]	aEnableEui64Filtering	TRUE to filter responses on EUI-64, FALSE otherwise.
[in]	aCallback	A pointer to a function called on receiving an MLE Discovery Response or scan completes.
[in]	aCallbackContext	A pointer to application-specific context.

Note

- A successful call to this function enables the rx-on-when-idle mode for the entire scan procedure.

Definition at line `285` of file `include/openthread/thread.h`

otThreadIsDiscoverInProgress

> bool otThreadIsDiscoverInProgress (otInstance *aInstance)

Determines if an MLE Thread Discovery is currently in progress.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `299` of file `include/openthread/thread.h`

otThreadSetJoinerAdvertisement

> otError otThreadSetJoinerAdvertisement (otInstance *aInstance, uint32_t aOui, const uint8_t *aAdvData, uint8_t aAdvDataLength)

Sets the Thread Joiner Advertisement when discovering Thread network.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aOui	The Vendor IEEE OUI value that will be included in the Joiner Advertisement. Only the least significant 3 bytes will be used, and the most significant byte will be ignored.
[in]	aAdvData	A pointer to the AdvData that will be included in the Joiner Advertisement.
[in]	aAdvDataLength	The length of AdvData in bytes.

Thread Joiner Advertisement is used to allow a Joiner to advertise its own application-specific information (such as Vendor ID, Product ID, Discriminator, etc.) via a newly-proposed Joiner Advertisement TLV, and to make this information available to Commissioners or Commissioner Candidates without human interaction.

Definition at line `318` of file `include/openthread/thread.h`

otThreadGetChildTimeout

```
uint32_t otThreadGetChildTimeout (otInstance *aInstance)
```

Gets the Thread Child Timeout (in seconds) used when operating in the Child role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Thread Child Timeout value in seconds.

See Also

- otThreadSetChildTimeout

Definition at line `335` of file `include/openthread/thread.h`

otThreadSetChildTimeout

```
void otThreadSetChildTimeout (otInstance *aInstance, uint32_t aTimeout)
```

Sets the Thread Child Timeout (in seconds) used when operating in the Child role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aTimeout	The timeout value in seconds.

See Also

- otThreadGetChildTimeout

Definition at line `346` of file `include/openthread/thread.h`

otThreadGetExtendedPanId

```
const otExtendedPanId * otThreadGetExtendedPanId (otInstance *aInstance)
```

Gets the IEEE 802.15.4 Extended PAN ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the IEEE 802.15.4 Extended PAN ID.

See Also

- otThreadSetExtendedPanId

Definition at line `358` of file `include/openthread/thread.h`

otThreadSetExtendedPanId

> otError otThreadSetExtendedPanId (otInstance *aInstance, const otExtendedPanId *aExtendedPanId)

Sets the IEEE 802.15.4 Extended PAN ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtendedPanId	A pointer to the IEEE 802.15.4 Extended PAN ID.

Note

- Can only be called while Thread protocols are disabled. A successful call to this function invalidates the Active and Pending Operational Datasets in non-volatile memory.

See Also

- otThreadGetExtendedPanId

Definition at line `376` of file `include/openthread/thread.h`

otThreadGetLeaderRloc

> otError otThreadGetLeaderRloc (otInstance *aInstance, otIp6Address *aLeaderRloc)

Returns a pointer to the Leader's RLOC.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aLeaderRloc	A pointer to the Leader's RLOC.

Definition at line `389` of file `include/openthread/thread.h`

otThreadGetLinkMode

> otLinkModeConfig otThreadGetLinkMode (otInstance *aInstance)

Get the MLE Link Mode configuration.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The MLE Link Mode configuration.

See Also

- otThreadSetLinkMode

Definition at line `401` of file `include/openthread/thread.h`

otThreadSetLinkMode

> otError otThreadSetLinkMode (otInstance *aInstance, otLinkModeConfig aConfig)

Set the MLE Link Mode configuration.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aConfig	A pointer to the Link Mode configuration.

See Also

- otThreadGetLinkMode

Definition at line `414` of file `include/openthread/thread.h`

otThreadGetNetworkKey

```
void otThreadGetNetworkKey (otInstance *aInstance, otNetworkKey *aNetworkKey)
```

Get the Thread Network Key.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aNetworkKey	A pointer to an `otNetworkKey` to return the Thread Network Key.

See Also

- otThreadSetNetworkKey

Definition at line `425` of file `include/openthread/thread.h`

otThreadGetNetworkKeyRef

```
otNetworkKeyRef otThreadGetNetworkKeyRef (otInstance *aInstance)
```

Get the `otNetworkKeyRef` for Thread Network Key.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Requires the build-time feature `OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE` to be enabled.

Returns

- Reference to the Thread Network Key stored in memory.

See Also

- otThreadSetNetworkKeyRef

Definition at line `439` of file `include/openthread/thread.h`

otThreadSetNetworkKey

```
otError otThreadSetNetworkKey (otInstance *aInstance, const otNetworkKey *aKey)
```

Set the Thread Network Key.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aKey	A pointer to a buffer containing the Thread Network Key.

Succeeds only when Thread protocols are disabled. A successful call to this function invalidates the Active and Pending Operational Datasets in non-volatile memory.

See Also

- otThreadGetNetworkKey

Definition at line 457 of file include/openthread/thread.h

otThreadSetNetworkKeyRef

```
otError otThreadSetNetworkKeyRef (otInstance *aInstance, otNetworkKeyRef aKeyRef)
```

Set the Thread Network Key as a otNetworkKeyRef.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aKeyRef	Reference to the Thread Network Key.

Succeeds only when Thread protocols are disabled. A successful call to this function invalidates the Active and Pending Operational Datasets in non-volatile memory.

Requires the build-time feature OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE to be enabled.

See Also

- otThreadGetNetworkKeyRef

Definition at line 477 of file include/openthread/thread.h

otThreadGetRloc

```
const otIp6Address * otThreadGetRloc (otInstance *aInstance)
```

Gets the Thread Routing Locator (RLOC) address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the Thread Routing Locator (RLOC) address.

Definition at line 487 of file include/openthread/thread.h

otThreadGetMeshLocalEid

```
const otIp6Address * otThreadGetMeshLocalEid (otInstance *aInstance)
```

Gets the Mesh Local EID address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the Mesh Local EID address.

Definition at line 497 of file include/openthread/thread.h

otThreadGetMeshLocalPrefix

```
const otMeshLocalPrefix * otThreadGetMeshLocalPrefix (otInstance *aInstance)
```

Returns a pointer to the Mesh Local Prefix.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the Mesh Local Prefix.

Definition at line 507 of file include/openthread/thread.h

otThreadSetMeshLocalPrefix

```
otError otThreadSetMeshLocalPrefix (otInstance *aInstance, const otMeshLocalPrefix *aMeshLocalPrefix)
```

Sets the Mesh Local Prefix.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMeshLocalPrefix	A pointer to the Mesh Local Prefix.

Succeeds only when Thread protocols are disabled. A successful call to this function invalidates the Active and Pending Operational Datasets in non-volatile memory.

Definition at line 523 of file include/openthread/thread.h

otThreadGetLinkLocalIp6Address

```
const otIp6Address * otThreadGetLinkLocalIp6Address (otInstance *aInstance)
```

Gets the Thread link-local IPv6 address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The Thread link local address is derived using IEEE802.15.4 Extended Address as Interface Identifier.

Returns

- A pointer to Thread link-local IPv6 address.

Definition at line 535 of file include/openthread/thread.h

otThreadGetLinkLocalAllThreadNodesMulticastAddress

```
const otIp6Address * otThreadGetLinkLocalAllThreadNodesMulticastAddress (otInstance *aInstance)
```

Gets the Thread Link-Local All Thread Nodes multicast address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The address is a link-local Unicast Prefix-Based Multicast Address [RFC 3306], with:

- flgs set to 3 (P = 1 and T = 1)
- scop set to 2
- plen set to 64
- network prefix set to the Mesh Local Prefix
- group ID set to 1

Returns

- A pointer to Thread Link-Local All Thread Nodes multicast address.

Definition at line 552 of file include/openthread/thread.h

otThreadGetRealmLocalAllThreadNodesMulticastAddress

```
const otIp6Address * otThreadGetRealmLocalAllThreadNodesMulticastAddress (otInstance *aInstance)
```

Gets the Thread Realm-Local All Thread Nodes multicast address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The address is a realm-local Unicast Prefix-Based Multicast Address [RFC 3306], with:

- flgs set to 3 (P = 1 and T = 1)
- scop set to 3
- plen set to 64
- network prefix set to the Mesh Local Prefix
- group ID set to 1

Returns

- A pointer to Thread Realm-Local All Thread Nodes multicast address.

Definition at line 569 of file include/openthread/thread.h

otThreadGetServiceAloc

```
otError otThreadGetServiceAloc (otInstance *aInstance, uint8_t aServiceId, otIp6Address *aServiceAloc)
```

Retrieves the Service ALOC for given Service ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[in]	aServiceId	Service ID to get ALOC for.
[out]	aServiceAloc	A pointer to output the Service ALOC. MUST NOT BE NULL.

Definition at line 581 of file include/openthread/thread.h

otThreadGetNetworkName

```
const char * otThreadGetNetworkName (otInstance *aInstance)
```

Get the Thread Network Name.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the Thread Network Name.

See Also

- otThreadSetNetworkName

Definition at line 593 of file include/openthread/thread.h

otThreadSetNetworkName

```
otError otThreadSetNetworkName (otInstance *aInstance, const char *aNetworkName)
```

Set the Thread Network Name.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aNetworkName	A pointer to the Thread Network Name.

Succeeds only when Thread protocols are disabled. A successful call to this function invalidates the Active and Pending Operational Datasets in non-volatile memory.

See Also

- otThreadGetNetworkName

Definition at line 611 of file include/openthread/thread.h

otThreadGetDomainName

```
const char * otThreadGetDomainName (otInstance *aInstance)
```

Gets the Thread Domain Name.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Note

- Available since Thread 1.2.

Returns

- A pointer to the Thread Domain Name.

See Also

- otThreadSetDomainName

Definition at line 625 of file include/openthread/thread.h

otThreadSetDomainName

> otError otThreadSetDomainName (otInstance *aInstance, const char *aDomainName)

Sets the Thread Domain Name.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDomainName	A pointer to the Thread Domain Name.

Only succeeds when Thread protocols are disabled.

Note

- Available since Thread 1.2.

See Also

- otThreadGetDomainName

Definition at line 641 of file include/openthread/thread.h

otThreadSetFixedDuaInterfaceIdentifier

> otError otThreadSetFixedDuaInterfaceIdentifier (otInstance *aInstance, const otIp6InterfaceIdentifier *aId)

Sets or clears the Interface Identifier manually specified for the Thread Domain Unicast Address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aId	A pointer to the Interface Identifier to set or NULL to clear.

Available when OPENTHREAD_CONFIG_DUA_ENABLE is enabled.

Note

- Only available since Thread 1.2.

See Also

- otThreadGetFixedDuaInterfaceIdentifier

Definition at line 658 of file include/openthread/thread.h

otThreadGetFixedDuaInterfaceIdentifier

> const otIp6InterfaceIdentifier * otThreadGetFixedDuaInterfaceIdentifier (otInstance *aInstance)

Gets the Interface Identifier manually specified for the Thread Domain Unicast Address.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Available when `OPENTHREAD_CONFIG_DUA_ENABLE` is enabled.

Note

- Only available since Thread 1.2.

Returns

- A pointer to the Interface Identifier which was set manually, or NULL if none was set.

See Also

- otThreadSetFixedDuaInterfaceIdentifier

Definition at line `674` of file `include/openthread/thread.h`

otThreadGetKeySequenceCounter

```
uint32_t otThreadGetKeySequenceCounter (otInstance *aInstance)
```

Gets the thrKeySequenceCounter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The thrKeySequenceCounter value.

See Also

- otThreadSetKeySequenceCounter

Definition at line `686` of file `include/openthread/thread.h`

otThreadSetKeySequenceCounter

```
void otThreadSetKeySequenceCounter (otInstance *aInstance, uint32_t aKeySequenceCounter)
```

Sets the thrKeySequenceCounter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aKeySequenceCounter	The thrKeySequenceCounter value.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

See Also

- otThreadGetKeySequenceCounter

Definition at line 700 of file include/openthread/thread.h

otThreadGetKeySwitchGuardTime

```
uint32_t otThreadGetKeySwitchGuardTime (otInstance *aInstance)
```

Gets the thrKeySwitchGuardTime (in hours).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The thrKeySwitchGuardTime value (in hours).

See Also

- otThreadSetKeySwitchGuardTime

Definition at line 712 of file include/openthread/thread.h

otThreadSetKeySwitchGuardTime

```
void otThreadSetKeySwitchGuardTime (otInstance *aInstance, uint32_t aKeySwitchGuardTime)
```

Sets the thrKeySwitchGuardTime (in hours).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aKeySwitchGuardTime	The thrKeySwitchGuardTime value (in hours).

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

See Also

- otThreadGetKeySwitchGuardTime

Definition at line 726 of file include/openthread/thread.h

otThreadBecomeDetached

```
otError otThreadBecomeDetached (otInstance *aInstance)
```

Detach from the Thread network.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 737 of file include/openthread/thread.h

otThreadBecomeChild

> otError otThreadBecomeChild (otInstance *aInstance)

Attempt to reattach as a child.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

Definition at line `751` of file `include/openthread/thread.h`

otThreadGetNextNeighborInfo

> otError otThreadGetNextNeighborInfo (otInstance *aInstance, otNeighborInfoIterator *aIterator, otNeighborInfo *aInfo)

Gets the next neighbor information.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the iterator context. To get the first neighbor entry it should be set to OT_NEIGHBOR_INFO_ITERATOR_INIT.
[out]	aInfo	A pointer to the neighbor information.

It is used to go through the entries of the neighbor table.

Definition at line `767` of file `include/openthread/thread.h`

otThreadGetDeviceRole

> otDeviceRole otThreadGetDeviceRole (otInstance *aInstance)

Get the device role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `781` of file `include/openthread/thread.h`

otThreadDeviceRoleToString

> const char * otThreadDeviceRoleToString (otDeviceRole aRole)

Convert the device role to human-readable string.

Parameters

[in]	aRole	The device role to convert.

Returns

A string representing `aRole`.

Definition at line `791` of file `include/openthread/thread.h`

otThreadGetLeaderData

```
otError otThreadGetLeaderData (otInstance *aInstance, otLeaderData *aLeaderData)
```

Get the Thread Leader Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aLeaderData	A pointer to where the leader data is placed.

Definition at line `803` of file `include/openthread/thread.h`

otThreadGetLeaderRouterId

```
uint8_t otThreadGetLeaderRouterId (otInstance *aInstance)
```

Get the Leader's Router ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Leader's Router ID.

Definition at line `813` of file `include/openthread/thread.h`

otThreadGetLeaderWeight

```
uint8_t otThreadGetLeaderWeight (otInstance *aInstance)
```

Get the Leader's Weight.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Leader's Weight.

Definition at line `823` of file `include/openthread/thread.h`

otThreadGetPartitionId

```
uint32_t otThreadGetPartitionId (otInstance *aInstance)
```

Get the Partition ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Partition ID.

Definition at line 833 of file include/openthread/thread.h

otThreadGetRloc16

```
uint16_t otThreadGetRloc16 (otInstance *aInstance)
```

Get the RLOC16.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The RLOC16.

Definition at line 843 of file include/openthread/thread.h

otThreadGetParentInfo

```
otError otThreadGetParentInfo (otInstance *aInstance, otRouterInfo *aParentInfo)
```

The function retrieves diagnostic information for a Thread Router as parent.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aParentInfo	A pointer to where the parent router information is placed.

Definition at line 852 of file include/openthread/thread.h

otThreadGetParentAverageRssi

```
otError otThreadGetParentAverageRssi (otInstance *aInstance, int8_t *aParentRssi)
```

The function retrieves the average RSSI for the Thread Parent.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aParentRssi	A pointer to where the parent RSSI should be placed.

Definition at line 861 of file include/openthread/thread.h

otThreadGetParentLastRssi

```
otError otThreadGetParentLastRssi (otInstance *aInstance, int8_t *aLastRssi)
```

The function retrieves the RSSI of the last packet from the Thread Parent.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aLastRssi	A pointer to where the last RSSI should be placed.

Definition at line `874` of file `include/openthread/thread.h`

otThreadSearchForBetterParent

```
otError otThreadSearchForBetterParent (otInstance *aInstance)
```

Starts the process for child to search for a better parent while staying attached to its current parent.

Parameters

N/A	aInstance	

Must be used when device is attached as a child.

Definition at line `885` of file `include/openthread/thread.h`

otThreadGetIp6Counters

```
const otIpCounters * otThreadGetIp6Counters (otInstance *aInstance)
```

Gets the IPv6 counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the IPv6 counters.

Definition at line `895` of file `include/openthread/thread.h`

otThreadResetIp6Counters

```
void otThreadResetIp6Counters (otInstance *aInstance)
```

Resets the IPv6 counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `903` of file `include/openthread/thread.h`

otThreadGetTimeInQueueHistogram

```
const uint32_t * otThreadGetTimeInQueueHistogram (otInstance *aInstance, uint16_t *aNumBins, uint32_t *aBinInterval)
```

Gets the time-in-queue histogram for messages in the TX queue.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aNumBins	Pointer to return the number of bins in histogram (array length).
[out]	aBinInterval	Pointer to return the histogram bin interval length in milliseconds.

Requires `OPENTHREAD_CONFIG_TX_QUEUE_STATISTICS_ENABLE`.

Histogram of the time-in-queue of messages in the transmit queue is collected. The time-in-queue is tracked for direct transmissions only and is measured as the duration from when a message is added to the transmit queue until it is passed to the MAC layer for transmission or dropped.

The histogram is returned as an array of `uint32_t` values with `aNumBins` entries. The first entry in the array (at index 0) represents the number of messages with a time-in-queue less than `aBinInterval`. The second entry represents the number of messages with a time-in-queue greater than or equal to `aBinInterval`, but less than `2 * aBinInterval`. And so on. The last entry represents the number of messages with time-in-queue greater than or equal to `(aNumBins - 1) * aBinInterval`.

The collected statistics can be reset by calling `otThreadResetTimeInQueueStat()`. The histogram information is collected since the OpenThread instance was initialized or since the last time statistics collection was reset by calling the `otThreadResetTimeInQueueStat()`.

Pointers `aNumBins` and `aBinInterval` MUST NOT be NULL.

Returns

- A pointer to an array of `aNumBins` entries representing the collected histogram info.

Definition at line `933` of file `include/openthread/thread.h`

otThreadGetMaxTimeInQueue

```
uint32_t otThreadGetMaxTimeInQueue (otInstance *aInstance)
```

Gets the maximum time-in-queue for messages in the TX queue.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Requires `OPENTHREAD_CONFIG_TX_QUEUE_STATISTICS_ENABLE`.

The time-in-queue is tracked for direct transmissions only and is measured as the duration from when a message is added to the transmit queue until it is passed to the MAC layer for transmission or dropped.

The collected statistics can be reset by calling `otThreadResetTimeInQueueStat()`.

Returns

- The maximum time-in-queue in milliseconds for all messages in the TX queue (so far).

Definition at line `950` of file `include/openthread/thread.h`

otThreadResetTimeInQueueStat

```
void otThreadResetTimeInQueueStat (otInstance *aInstance)
```

Resets the TX queue time-in-queue statistics.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Requires `OPENTHREAD_CONFIG_TX_QUEUE_STATISTICS_ENABLE`.

Definition at line `960` of file `include/openthread/thread.h`

otThreadGetMleCounters

```
const otMleCounters * otThreadGetMleCounters (otInstance *aInstance)
```

Gets the Thread MLE counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- A pointer to the Thread MLE counters.

Definition at line `970` of file `include/openthread/thread.h`

otThreadResetMleCounters

```
void otThreadResetMleCounters (otInstance *aInstance)
```

Resets the Thread MLE counters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `978` of file `include/openthread/thread.h`

otThreadRegisterParentResponseCallback

```
void otThreadRegisterParentResponseCallback (otInstance *aInstance, otThreadParentResponseCallback aCallback, void *aContext)
```

Registers a callback to receive MLE Parent Response data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function that is called upon receiving an MLE Parent Response message.
[in]	aContext	A pointer to callback client-specific context.

Requires `OPENTHREAD_CONFIG_MLE_PARENT_RESPONSE_CALLBACK_API_ENABLE`.

Definition at line `1001` of file `include/openthread/thread.h`

otThreadSetDiscoveryRequestCallback

> void otThreadSetDiscoveryRequestCallback (otInstance *aInstance, otThreadDiscoveryRequestCallback aCallback, void *aContext)

Sets a callback to receive MLE Discovery Request data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function that is called upon receiving an MLE Discovery Request message.
[in]	aContext	A pointer to callback application-specific context.

Definition at line `1033` of file `include/openthread/thread.h`

otThreadLocateAnycastDestination

> otError otThreadLocateAnycastDestination (otInstance *aInstance, const otIp6Address *aAnycastAddress, otThreadAnycastLocatorCallback aCallback, void *aContext)

Requests the closest destination of a given anycast address to be located.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aAnycastAddress	The anycast address to locate. MUST NOT be NULL.
[in]	aCallback	The callback function to report the result.
[in]	aContext	An arbitrary context used with `aCallback` .

Is only available when `OPENTHREAD_CONFIG_TMF_ANYCAST_LOCATOR_ENABLE` is enabled.

If a previous request is ongoing, a subsequent call to this function will cancel and replace the earlier request.

Definition at line `1072` of file `include/openthread/thread.h`

otThreadIsAnycastLocateInProgress

> bool otThreadIsAnycastLocateInProgress (otInstance *aInstance)

Indicates whether an anycast locate request is currently in progress.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Is only available when `OPENTHREAD_CONFIG_TMF_ANYCAST_LOCATOR_ENABLE` is enabled.

Returns

- TRUE if an anycast locate request is currently in progress, FALSE otherwise.

Definition at line `1087` of file `include/openthread/thread.h`

otThreadSendAddressNotification

> void otThreadSendAddressNotification (otInstance *aInstance, otIp6Address *aDestination, otIp6Address *aTarget, otIp6InterfaceIdentifier *aMlIid)

Sends a Proactive Address Notification (ADDR_NTF.ntf) message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDestination	The destination to send the ADDR_NTF.ntf message.
[in]	aTarget	The target address of the ADDR_NTF.ntf message.
[in]	aMlIid	The ML-IID of the ADDR_NTF.ntf message.

Is only available when `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE` is enabled.

Definition at line `1100` of file `include/openthread/thread.h`

otThreadSendProactiveBackboneNotification

```
otError otThreadSendProactiveBackboneNotification (otInstance *aInstance, otIp6Address *aTarget,
otIp6InterfaceIdentifier *aMlIid, uint32_t aTimeSinceLastTransaction)
```

Sends a Proactive Backbone Notification (PRO_BB.ntf) message on the Backbone link.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aTarget	The target address of the PRO_BB.ntf message.
[in]	aMlIid	The ML-IID of the PRO_BB.ntf message.
[in]	aTimeSinceLastTransaction	Time since last transaction (in seconds).

Is only available when `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE` is enabled.

Definition at line `1119` of file `include/openthread/thread.h`

otThreadDetachGracefully

```
otError otThreadDetachGracefully (otInstance *aInstance, otDetachGracefullyCallback aCallback, void *aContext)
```

Notifies other nodes in the network (if any) and then stops Thread protocol operation.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function that is called upon finishing detaching.
[in]	aContext	A pointer to callback application-specific context.

It sends an Address Release if it's a router, or sets its child timeout to 0 if it's a child.

Definition at line `1137` of file `include/openthread/thread.h`

otConvertDurationInSecondsToString

```
void otConvertDurationInSecondsToString (uint32_t aDuration, char *aBuffer, uint16_t aSize)
```

Converts an `uint32_t` duration (in seconds) to a human-readable string.

Parameters

[in]	aDuration	A duration interval in seconds.
[out]	aBuffer	A pointer to a char array to output the string.
[in]	aSize	The size of `aBuffer` (in bytes). Recommended to use `OT_DURATION_STRING_SIZE`.

Requires `OPENTHREAD_CONFIG_UPTIME_ENABLE` to be enabled.

The string follows the format "<hh>:<mm>:<ss>" for hours, minutes, seconds (if duration is shorter than one day) or "<dd>d.<hh>:<mm>:<ss>" (if longer than a day).

If the resulting string does not fit in `aBuffer` (within its `aSize` characters), the string will be truncated but the outputted string is always null-terminated.

Is intended for use with `mAge` or `mConnectionTime` in `otNeighborInfo` or `otChildInfo` structures.

Definition at line `1159` of file `include/openthread/thread.h`

Macro Definition Documentation

OT_NETWORK_DATA_ITERATOR_INIT

```
#define OT_NETWORK_DATA_ITERATOR_INIT
```

Value:

```
0
```

Value to initialize `otNetworkDataIterator`.

Definition at line `52` of file `include/openthread/netdata.h`

OT_SERVICE_DATA_MAX_SIZE

```
#define OT_SERVICE_DATA_MAX_SIZE
```

Value:

```
252
```

Max size of Service Data in bytes.

Definition at line `112` of file `include/openthread/netdata.h`

OT_SERVER_DATA_MAX_SIZE

```
#define OT_SERVER_DATA_MAX_SIZE
```

Value:

```
248
```

Max size of Server Data in bytes. Theoretical limit, practically much lower.

Definition at line `113` of file `include/openthread/netdata.h`

OT_NETWORK_DIAGNOSTIC_TYPELIST_MAX_ENTRIES

```
#define OT_NETWORK_DIAGNOSTIC_TYPELIST_MAX_ENTRIES
```

Value:

```
19
```

Maximum Number of Network Diagnostic TLV Types to Request or Reset.

Definition at line `55` of file `include/openthread/netdiag.h`

OT_NETWORK_DIAGNOSTIC_CHILD_TABLE_ENTRY_SIZE

```
#define OT_NETWORK_DIAGNOSTIC_CHILD_TABLE_ENTRY_SIZE
```

Value:

```
3
```

Size of Network Diagnostic Child Table entry.

Definition at line `60` of file `include/openthread/netdiag.h`

OT_NETWORK_DIAGNOSTIC_ITERATOR_INIT

```
#define OT_NETWORK_DIAGNOSTIC_ITERATOR_INIT
```

Value:

```
0
```

Initializer for otNetworkDiagIterator.

Definition at line `65` of file `include/openthread/netdiag.h`

OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_NAME_TLV_LENGTH

```
#define OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_NAME_TLV_LENGTH
```

Value:

```
32
```

Max length of Vendor Name TLV.

Definition at line `99` of file `include/openthread/netdiag.h`

OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_MODEL_TLV_LENGTH

```
#define OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_MODEL_TLV_LENGTH
```

Value:

```
32
```

Max length of Vendor Model TLV.

Definition at line 100 of file include/openthread/netdiag.h

OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_SW_VERSION_TLV_LENGTH

```
#define OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_SW_VERSION_TLV_LENGTH
```

Value:

```
16
```

Max length of Vendor SW Version TLV.

Definition at line 101 of file include/openthread/netdiag.h

OT_NETWORK_DIAGNOSTIC_MAX_THREAD_STACK_VERSION_TLV_LENGTH

```
#define OT_NETWORK_DIAGNOSTIC_MAX_THREAD_STACK_VERSION_TLV_LENGTH
```

Value:

```
64
```

Max length of Thread Stack Version TLV.

Definition at line 102 of file include/openthread/netdiag.h

OT_NETWORK_BASE_TLV_MAX_LENGTH

```
#define OT_NETWORK_BASE_TLV_MAX_LENGTH
```

Value:

```
254
```

Maximum value length of Thread Base TLV.

Definition at line 59 of file include/openthread/thread.h

OT_NETWORK_MAX_ROUTER_ID

```
#define OT_NETWORK_MAX_ROUTER_ID
```

Value:

```
62
```

Maximum Router ID.

Definition at line 61 of file include/openthread/thread.h

OT_NEIGHBOR_INFO_ITERATOR_INIT

```
#define OT_NEIGHBOR_INFO_ITERATOR_INIT
```

Value:

```
0
```

Initializer for otNeighborInfoIterator.

Definition at line 111 of file include/openthread/thread.h

OT_JOINER_ADVDATA_MAX_LENGTH

```
#define OT_JOINER_ADVDATA_MAX_LENGTH
```

Value:

```
64
```

Maximum AdvData Length of Joiner Advertisement.

Definition at line 323 of file include/openthread/thread.h

OT_DURATION_STRING_SIZE

```
#define OT_DURATION_STRING_SIZE
```

Value:

```
21
```

Recommended size for string representation of uint32_t duration in seconds.

Definition at line 1139 of file include/openthread/thread.h

otBorderRouterConfig

Represents a Border Router configuration.

Public Attributes

otIp6Prefix	mPrefix The IPv6 prefix.
signed int	mPreference A 2-bit signed int preference (`OT_ROUTE_PREFERENCE_*` values).
bool	mPreferred Whether prefix is preferred.
bool	mSlaac Whether prefix can be used for address auto-configuration (SLAAC).
bool	mDhcp Whether border router is DHCPv6 Agent.
bool	mConfigure Whether DHCPv6 Agent supplying other config data.
bool	mDefaultRoute Whether border router is a default router for prefix.
bool	mOnMesh Whether this prefix is considered on-mesh.
bool	mStable Whether this configuration is considered Stable Network Data.
bool	mNdDns Whether this border router can supply DNS information via ND.
bool	mDp Whether prefix is a Thread Domain Prefix (added since Thread 1.2).
uint16_t	mRloc16 The border router's RLOC16 (value ignored on config add).

Public Attribute Documentation

mPrefix

> otIp6Prefix otBorderRouterConfig::mPrefix

The IPv6 prefix.

Definition at line `61` of file `include/openthread/netdata.h`

mPreference

signed int otBorderRouterConfig::mPreference

A 2-bit signed int preference (`OT_ROUTE_PREFERENCE_*` values).

Definition at line `62` of file `include/openthread/netdata.h`

mPreferred

bool otBorderRouterConfig::mPreferred

Whether prefix is preferred.

Definition at line `63` of file `include/openthread/netdata.h`

mSlaac

bool otBorderRouterConfig::mSlaac

Whether prefix can be used for address auto-configuration (SLAAC).

Definition at line `64` of file `include/openthread/netdata.h`

mDhcp

bool otBorderRouterConfig::mDhcp

Whether border router is DHCPv6 Agent.

Definition at line `65` of file `include/openthread/netdata.h`

mConfigure

bool otBorderRouterConfig::mConfigure

Whether DHCPv6 Agent supplying other config data.

Definition at line `66` of file `include/openthread/netdata.h`

mDefaultRoute

bool otBorderRouterConfig::mDefaultRoute

Whether border router is a default router for prefix.

Definition at line `67` of file `include/openthread/netdata.h`

mOnMesh

bool otBorderRouterConfig::mOnMesh

Whether this prefix is considered on-mesh.

Definition at line 68 of file include/openthread/netdata.h

mStable

bool otBorderRouterConfig::mStable

Whether this configuration is considered Stable Network Data.

Definition at line 69 of file include/openthread/netdata.h

mNdDns

bool otBorderRouterConfig::mNdDns

Whether this border router can supply DNS information via ND.

Definition at line 70 of file include/openthread/netdata.h

mDp

bool otBorderRouterConfig::mDp

Whether prefix is a Thread Domain Prefix (added since Thread 1.2).

Definition at line 71 of file include/openthread/netdata.h

mRloc16

uint16_t otBorderRouterConfig::mRloc16

The border router's RLOC16 (value ignored on config add).

Definition at line 72 of file include/openthread/netdata.h

otLowpanContextInfo

Represents 6LoWPAN Context ID information associated with a prefix in Network Data.

Public Attributes

uint8_t	mContextId The 6LoWPAN Context ID.
bool	mCompressFlag The compress flag.
otIp6Prefix	mPrefix The associated IPv6 prefix.

Public Attribute Documentation

mContextId

```
uint8_t otLowpanContextInfo::mContextId
```

The 6LoWPAN Context ID.

Definition at line 81 of file include/openthread/netdata.h

mCompressFlag

```
bool otLowpanContextInfo::mCompressFlag
```

The compress flag.

Definition at line 82 of file include/openthread/netdata.h

mPrefix

```
otIp6Prefix otLowpanContextInfo::mPrefix
```

The associated IPv6 prefix.

Definition at line 83 of file include/openthread/netdata.h

otExternalRouteConfig

Represents an External Route configuration.

Public Attributes

otIp6Prefix	mPrefix The IPv6 prefix.
uint16_t	mRloc16 The border router's RLOC16 (value ignored on config add).
signed int	mPreference A 2-bit signed int preference (`OT_ROUTE_PREFERENCE_*` values).
bool	mNat64 Whether this is a NAT64 prefix.
bool	mStable Whether this configuration is considered Stable Network Data.
bool	mNextHopIsThisDevice Whether the next hop is this device (value ignored on config add).
bool	mAdvPio Whether or not BR is advertising a ULA prefix in PIO (AP flag).

Public Attribute Documentation

mPrefix

> otIp6Prefix otExternalRouteConfig::mPrefix

The IPv6 prefix.

Definition at line `92` of file `include/openthread/netdata.h`

mRloc16

> uint16_t otExternalRouteConfig::mRloc16

The border router's RLOC16 (value ignored on config add).

Definition at line `93` of file `include/openthread/netdata.h`

mPreference

> signed int otExternalRouteConfig::mPreference

A 2-bit signed int preference (`OT_ROUTE_PREFERENCE_*` values).

Definition at line 94 of file include/openthread/netdata.h

mNat64

```
bool otExternalRouteConfig::mNat64
```

Whether this is a NAT64 prefix.

Definition at line 95 of file include/openthread/netdata.h

mStable

```
bool otExternalRouteConfig::mStable
```

Whether this configuration is considered Stable Network Data.

Definition at line 96 of file include/openthread/netdata.h

mNextHopIsThisDevice

```
bool otExternalRouteConfig::mNextHopIsThisDevice
```

Whether the next hop is this device (value ignored on config add).

Definition at line 97 of file include/openthread/netdata.h

mAdvPio

```
bool otExternalRouteConfig::mAdvPio
```

Whether or not BR is advertising a ULA prefix in PIO (AP flag).

Definition at line 98 of file include/openthread/netdata.h

otServerConfig

Represents a Server configuration.

Public Attributes

bool	mStable Whether this config is considered Stable Network Data.
uint8_t	mServerDataLength Length of server data.
uint8_t	mServerData Server data bytes.
uint16_t	mRloc16 The Server RLOC16.

Public Attribute Documentation

mStable

```
bool otServerConfig::mStable
```

Whether this config is considered Stable Network Data.

Definition at line 121 of file include/openthread/netdata.h

mServerDataLength

```
uint8_t otServerConfig::mServerDataLength
```

Length of server data.

Definition at line 122 of file include/openthread/netdata.h

mServerData

```
uint8_t otServerConfig::mServerData[OT_SERVER_DATA_MAX_SIZE]
```

Server data bytes.

Definition at line 123 of file include/openthread/netdata.h

mRloc16

```
uint16_t otServerConfig::mRloc16
```

The Server RLOC16.

Definition at line 124 of file include/openthread/netdata.h

otServiceConfig

Represents a Service configuration.

Public Attributes

uint8_t	mServiceId Service ID (when iterating over the Network Data).
uint32_t	mEnterpriseNumber IANA Enterprise Number.
uint8_t	mServiceDataLength Length of service data.
uint8_t	mServiceData Service data bytes.
otServerConfig	mServerConfig The Server configuration.

Public Attribute Documentation

mServiceId

```
uint8_t otServiceConfig::mServiceId
```

Service ID (when iterating over the Network Data).

Definition at line 133 of file include/openthread/netdata.h

mEnterpriseNumber

```
uint32_t otServiceConfig::mEnterpriseNumber
```

IANA Enterprise Number.

Definition at line 134 of file include/openthread/netdata.h

mServiceDataLength

```
uint8_t otServiceConfig::mServiceDataLength
```

Length of service data.

Definition at line 135 of file include/openthread/netdata.h

mServiceData

```
uint8_t otServiceConfig::mServiceData[OT_SERVICE_DATA_MAX_SIZE]
```

Service data bytes.

Definition at line 136 of file include/openthread/netdata.h

mServerConfig

```
otServerConfig otServiceConfig::mServerConfig
```

The Server configuration.

Definition at line 137 of file include/openthread/netdata.h

otNetworkDiagConnectivity

Represents a Network Diagnostic Connectivity value.

Public Attributes

int8_t	mParentPriority The priority of the sender as a parent.
uint8_t	mLinkQuality3 The number of neighboring devices with which the sender shares a link of quality 3.
uint8_t	mLinkQuality2 The number of neighboring devices with which the sender shares a link of quality 2.
uint8_t	mLinkQuality1 The number of neighboring devices with which the sender shares a link of quality 1.
uint8_t	mLeaderCost The sender's routing cost to the Leader.
uint8_t	mIdSequence The most recent ID sequence number received by the sender.
uint8_t	mActiveRouters The number of active Routers in the sender's Thread Network Partition.
uint16_t	mSedBufferSize The guaranteed buffer capacity in octets for all IPv6 datagrams destined to a given SED.
uint8_t	mSedDatagramCount The guaranteed queue capacity in number of IPv6 datagrams destined to a given SED.

Public Attribute Documentation

mParentPriority

```
int8_t otNetworkDiagConnectivity::mParentPriority
```

The priority of the sender as a parent.

Definition at line 115 of file include/openthread/netdiag.h

mLinkQuality3

```
uint8_t otNetworkDiagConnectivity::mLinkQuality3
```

The number of neighboring devices with which the sender shares a link of quality 3.

Definition at line 120 of file include/openthread/netdiag.h

mLinkQuality2

```
uint8_t otNetworkDiagConnectivity::mLinkQuality2
```

The number of neighboring devices with which the sender shares a link of quality 2.

Definition at line 125 of file include/openthread/netdiag.h

mLinkQuality1

```
uint8_t otNetworkDiagConnectivity::mLinkQuality1
```

The number of neighboring devices with which the sender shares a link of quality 1.

Definition at line 130 of file include/openthread/netdiag.h

mLeaderCost

```
uint8_t otNetworkDiagConnectivity::mLeaderCost
```

The sender's routing cost to the Leader.

Definition at line 135 of file include/openthread/netdiag.h

mIdSequence

```
uint8_t otNetworkDiagConnectivity::mIdSequence
```

The most recent ID sequence number received by the sender.

Definition at line 140 of file include/openthread/netdiag.h

mActiveRouters

```
uint8_t otNetworkDiagConnectivity::mActiveRouters
```

The number of active Routers in the sender's Thread Network Partition.

Definition at line 145 of file include/openthread/netdiag.h

mSedBufferSize

```
uint16_t otNetworkDiagConnectivity::mSedBufferSize
```

The guaranteed buffer capacity in octets for all IPv6 datagrams destined to a given SED.

Optional.

Definition at line 150 of file include/openthread/netdiag.h

mSedDatagramCount

> uint8_t otNetworkDiagConnectivity::mSedDatagramCount

The guaranteed queue capacity in number of IPv6 datagrams destined to a given SED.

Optional.

Definition at line 155 of file include/openthread/netdiag.h

otNetworkDiagRouteData

Represents a Network Diagnostic Route data.

Public Attributes

uint8_t	mRouterId The Assigned Router ID.
uint8_t	mLinkQualityOut Link Quality Out.
uint8_t	mLinkQualityIn Link Quality In.
uint8_t	mRouteCost Routing Cost. Infinite routing cost is represented by value 0.

Public Attribute Documentation

mRouterId

```
uint8_t otNetworkDiagRouteData::mRouterId
```

The Assigned Router ID.

Definition at line 164 of file include/openthread/netdiag.h

mLinkQualityOut

```
uint8_t otNetworkDiagRouteData::mLinkQualityOut
```

Link Quality Out.

Definition at line 165 of file include/openthread/netdiag.h

mLinkQualityIn

```
uint8_t otNetworkDiagRouteData::mLinkQualityIn
```

Link Quality In.

Definition at line 166 of file include/openthread/netdiag.h

mRouteCost

```
uint8_t otNetworkDiagRouteData::mRouteCost
```

Routing Cost. Infinite routing cost is represented by value 0.

Definition at line 167 of file include/openthread/netdiag.h

otNetworkDiagRoute

Represents a Network Diagnostic Route TLV value.

Public Attributes

uint8_t	mIdSequence The sequence number associated with the set of Router ID assignments in mRouteData.
uint8_t	mRouteCount Number of elements in mRouteData.
otNetworkDiagRouteData	mRouteData Link Quality and Routing Cost data.

Public Attribute Documentation

mIdSequence

```
uint8_t otNetworkDiagRoute::mIdSequence
```

The sequence number associated with the set of Router ID assignments in mRouteData.

Definition at line 179 of file include/openthread/netdiag.h

mRouteCount

```
uint8_t otNetworkDiagRoute::mRouteCount
```

Number of elements in mRouteData.

Definition at line 184 of file include/openthread/netdiag.h

mRouteData

```
otNetworkDiagRouteData otNetworkDiagRoute::mRouteData[OT_NETWORK_MAX_ROUTER_ID+1]
```

Link Quality and Routing Cost data.

Definition at line 189 of file include/openthread/netdiag.h

otNetworkDiagMacCounters

Represents a Network Diagnostic Mac Counters value.

See RFC 2863 for definitions of member fields.

Public Attributes

uint32_t	mIfInUnknownProtos
uint32_t	mIfInErrors
uint32_t	mIfOutErrors
uint32_t	mIfInUcastPkts
uint32_t	mIfInBroadcastPkts
uint32_t	mIfInDiscards
uint32_t	mIfOutUcastPkts
uint32_t	mIfOutBroadcastPkts
uint32_t	mIfOutDiscards

Public Attribute Documentation

mIfInUnknownProtos

```
uint32_t otNetworkDiagMacCounters::mIfInUnknownProtos
```

Definition at line 200 of file include/openthread/netdiag.h

mIfInErrors

```
uint32_t otNetworkDiagMacCounters::mIfInErrors
```

Definition at line 201 of file include/openthread/netdiag.h

mIfOutErrors

```
uint32_t otNetworkDiagMacCounters::mIfOutErrors
```

Definition at line 202 of file include/openthread/netdiag.h

mIfInUcastPkts

```
uint32_t otNetworkDiagMacCounters::mIfInUcastPkts
```

Definition at line 203 of file include/openthread/netdiag.h

mIfInBroadcastPkts

```
uint32_t otNetworkDiagMacCounters::mIfInBroadcastPkts
```

Definition at line 204 of file include/openthread/netdiag.h

mIfInDiscards

```
uint32_t otNetworkDiagMacCounters::mIfInDiscards
```

Definition at line 205 of file include/openthread/netdiag.h

mIfOutUcastPkts

```
uint32_t otNetworkDiagMacCounters::mIfOutUcastPkts
```

Definition at line 206 of file include/openthread/netdiag.h

mIfOutBroadcastPkts

```
uint32_t otNetworkDiagMacCounters::mIfOutBroadcastPkts
```

Definition at line 207 of file include/openthread/netdiag.h

mIfOutDiscards

```
uint32_t otNetworkDiagMacCounters::mIfOutDiscards
```

Definition at line 208 of file include/openthread/netdiag.h

otNetworkDiagMleCounters

Represents a Network Diagnostics MLE Counters value.

Public Attributes

uint16_t	mDisabledRole Number of times device entered disabled role.
uint16_t	mDetachedRole Number of times device entered detached role.
uint16_t	mChildRole Number of times device entered child role.
uint16_t	mRouterRole Number of times device entered router role.
uint16_t	mLeaderRole Number of times device entered leader role.
uint16_t	mAttachAttempts Number of attach attempts while device was detached.
uint16_t	mPartitionIdChanges Number of changes to partition ID.
uint16_t	mBetterPartitionAttachAttempts Number of attempts to attach to a better partition.
uint16_t	mParentChanges Number of time device changed its parent.
uint64_t	mTrackedTime Milliseconds tracked by next counters (zero if not supported).
uint64_t	mDisabledTime Milliseconds device has been in disabled role.
uint64_t	mDetachedTime Milliseconds device has been in detached role.
uint64_t	mChildTime Milliseconds device has been in child role.
uint64_t	mRouterTime Milliseconds device has been in router role.
uint64_t	mLeaderTime Milliseconds device has been in leader role.

Public Attribute Documentation

mDisabledRole

> uint16_t otNetworkDiagMleCounters::mDisabledRole

Number of times device entered disabled role.

Definition at line 217 of file include/openthread/netdiag.h

mDetachedRole

```
uint16_t otNetworkDiagMleCounters::mDetachedRole
```

Number of times device entered detached role.

Definition at line 218 of file include/openthread/netdiag.h

mChildRole

```
uint16_t otNetworkDiagMleCounters::mChildRole
```

Number of times device entered child role.

Definition at line 219 of file include/openthread/netdiag.h

mRouterRole

```
uint16_t otNetworkDiagMleCounters::mRouterRole
```

Number of times device entered router role.

Definition at line 220 of file include/openthread/netdiag.h

mLeaderRole

```
uint16_t otNetworkDiagMleCounters::mLeaderRole
```

Number of times device entered leader role.

Definition at line 221 of file include/openthread/netdiag.h

mAttachAttempts

```
uint16_t otNetworkDiagMleCounters::mAttachAttempts
```

Number of attach attempts while device was detached.

Definition at line 222 of file include/openthread/netdiag.h

mPartitionIdChanges

```
uint16_t otNetworkDiagMleCounters::mPartitionIdChanges
```

Number of changes to partition ID.

Definition at line 223 of file include/openthread/netdiag.h

mBetterPartitionAttachAttempts

> uint16_t otNetworkDiagMleCounters::mBetterPartitionAttachAttempts

Number of attempts to attach to a better partition.

Definition at line 224 of file include/openthread/netdiag.h

mParentChanges

> uint16_t otNetworkDiagMleCounters::mParentChanges

Number of time device changed its parent.

Definition at line 225 of file include/openthread/netdiag.h

mTrackedTime

> uint64_t otNetworkDiagMleCounters::mTrackedTime

Milliseconds tracked by next counters (zero if not supported).

Definition at line 226 of file include/openthread/netdiag.h

mDisabledTime

> uint64_t otNetworkDiagMleCounters::mDisabledTime

Milliseconds device has been in disabled role.

Definition at line 227 of file include/openthread/netdiag.h

mDetachedTime

> uint64_t otNetworkDiagMleCounters::mDetachedTime

Milliseconds device has been in detached role.

Definition at line 228 of file include/openthread/netdiag.h

mChildTime

> uint64_t otNetworkDiagMleCounters::mChildTime

Milliseconds device has been in child role.

Definition at line 229 of file include/openthread/netdiag.h

mRouterTime

```
uint64_t otNetworkDiagMleCounters::mRouterTime
```

Milliseconds device has been in router role.

Definition at line 230 of file include/openthread/netdiag.h

mLeaderTime

```
uint64_t otNetworkDiagMleCounters::mLeaderTime
```

Milliseconds device has been in leader role.

Definition at line 231 of file include/openthread/netdiag.h

otNetworkDiagChildEntry

Represents a Network Diagnostic Child Table Entry.

Public Attributes

uint16_t	mTimeout Expected poll time expressed as 2^(Timeout-4) seconds.
uint8_t	mLinkQuality Link Quality In value in [0,3].
uint16_t	mChildId Child ID from which an RLOC can be generated.
otLinkModeConfig	mMode Link mode bits.

Public Attribute Documentation

mTimeout

```
uint16_t otNetworkDiagChildEntry::mTimeout
```

Expected poll time expressed as 2^(Timeout-4) seconds.

Definition at line 243 of file include/openthread/netdiag.h

mLinkQuality

```
uint8_t otNetworkDiagChildEntry::mLinkQuality
```

Link Quality In value in [0,3].

Value 0 indicates that sender does not support the feature to provide link quality info.

Definition at line 251 of file include/openthread/netdiag.h

mChildId

```
uint16_t otNetworkDiagChildEntry::mChildId
```

Child ID from which an RLOC can be generated.

Definition at line 256 of file include/openthread/netdiag.h

mMode

```
otLinkModeConfig otNetworkDiagChildEntry::mMode
```

Link mode bits.

Definition at line 261 of file include/openthread/netdiag.h

otNetworkDiagTlv

Represents a Network Diagnostic TLV.

Public Attributes

uint8_t	mType The Network Diagnostic TLV type.
otExtAddress	mExtAddress
uint16_t	mAddr16
otLinkModeConfig	mMode
uint32_t	mTimeout
otNetworkDiagConnectivity	mConnectivity
otNetworkDiagRoute	mRoute
otLeaderData	mLeaderData
otNetworkDiagMacCounters	mMacCounters
otNetworkDiagMleCounters	mMleCounters
uint8_t	mBatteryLevel
uint16_t	mSupplyVoltage
uint32_t	mMaxChildTimeout
uint16_t	mVersion
char	mVendorName
char	mVendorModel
char	mVendorSwVersion
char	mThreadStackVersion
uint8_t	mCount
uint8_t	m8
struct otNetworkDiagTlv::@6::@7	mNetworkData
otIp6Address	mList
struct otNetworkDiagTlv	

::@6::@8	mIp6AddrList
otNetworkDiagChildEntry	mTable
struct otNetworkDiagTlv ::@6::@9	mChildTable
struct otNetworkDiagTlv ::@6::@10	mChannelPages
union otNetworkDiagTlv ::@6	mData

Public Attribute Documentation

mType

```
uint8_t otNetworkDiagTlv::mType
```

The Network Diagnostic TLV type.

Definition at line 273 of file include/openthread/netdiag.h

mExtAddress

```
otExtAddress otNetworkDiagTlv::mExtAddress
```

Definition at line 277 of file include/openthread/netdiag.h

mAddr16

```
uint16_t otNetworkDiagTlv::mAddr16
```

Definition at line 278 of file include/openthread/netdiag.h

mMode

```
otLinkModeConfig otNetworkDiagTlv::mMode
```

Definition at line 279 of file include/openthread/netdiag.h

mTimeout

```
uint32_t otNetworkDiagTlv::mTimeout
```

Definition at line 280 of file include/openthread/netdiag.h

mConnectivity

> otNetworkDiagConnectivity otNetworkDiagTlv::mConnectivity

Definition at line 281 of file include/openthread/netdiag.h

mRoute

> otNetworkDiagRoute otNetworkDiagTlv::mRoute

Definition at line 282 of file include/openthread/netdiag.h

mLeaderData

> otLeaderData otNetworkDiagTlv::mLeaderData

Definition at line 283 of file include/openthread/netdiag.h

mMacCounters

> otNetworkDiagMacCounters otNetworkDiagTlv::mMacCounters

Definition at line 284 of file include/openthread/netdiag.h

mMleCounters

> otNetworkDiagMleCounters otNetworkDiagTlv::mMleCounters

Definition at line 285 of file include/openthread/netdiag.h

mBatteryLevel

> uint8_t otNetworkDiagTlv::mBatteryLevel

Definition at line 286 of file include/openthread/netdiag.h

mSupplyVoltage

> uint16_t otNetworkDiagTlv::mSupplyVoltage

Definition at line 287 of file include/openthread/netdiag.h

mMaxChildTimeout

uint32_t otNetworkDiagTlv::mMaxChildTimeout

Definition at line 288 of file include/openthread/netdiag.h

mVersion

uint16_t otNetworkDiagTlv::mVersion

Definition at line 289 of file include/openthread/netdiag.h

mVendorName

char otNetworkDiagTlv::mVendorName[OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_NAME_TLV_LENGTH+1]

Definition at line 290 of file include/openthread/netdiag.h

mVendorModel

char otNetworkDiagTlv::mVendorModel[OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_MODEL_TLV_LENGTH+1]

Definition at line 291 of file include/openthread/netdiag.h

mVendorSwVersion

char otNetworkDiagTlv::mVendorSwVersion[OT_NETWORK_DIAGNOSTIC_MAX_VENDOR_SW_VERSION_TLV_LENGTH+1]

Definition at line 292 of file include/openthread/netdiag.h

mThreadStackVersion

char otNetworkDiagTlv::mThreadStackVersion[OT_NETWORK_DIAGNOSTIC_MAX_THREAD_STACK_VERSION_TLV_LENGTH+1]

Definition at line 293 of file include/openthread/netdiag.h

mCount

uint8_t otNetworkDiagTlv::mCount

Definition at line 296 of file include/openthread/netdiag.h

m8

uint8_t otNetworkDiagTlv::m8[OT_NETWORK_BASE_TLV_MAX_LENGTH]

Definition at line 297 of file include/openthread/netdiag.h

mNetworkData

```
struct otNetworkDiagTlv::@6::@7 otNetworkDiagTlv::mNetworkData
```

Definition at line 298 of file include/openthread/netdiag.h

mList

```
otIp6Address otNetworkDiagTlv::mList[OT_NETWORK_BASE_TLV_MAX_LENGTH/OT_IP6_ADDRESS_SIZE]
```

Definition at line 302 of file include/openthread/netdiag.h

mIp6AddrList

```
struct otNetworkDiagTlv::@6::@8 otNetworkDiagTlv::mIp6AddrList
```

Definition at line 303 of file include/openthread/netdiag.h

mTable

```
otNetworkDiagChildEntry
otNetworkDiagTlv::mTable[OT_NETWORK_BASE_TLV_MAX_LENGTH/OT_NETWORK_DIAGNOSTIC_CHILD_TABLE_ENTRY_SIZE]
```

Definition at line 308 of file include/openthread/netdiag.h

mChildTable

```
struct otNetworkDiagTlv::@6::@9 otNetworkDiagTlv::mChildTable
```

Definition at line 309 of file include/openthread/netdiag.h

mChannelPages

```
struct otNetworkDiagTlv::@6::@10 otNetworkDiagTlv::mChannelPages
```

Definition at line 314 of file include/openthread/netdiag.h

mData

```
union otNetworkDiagTlv::@6 otNetworkDiagTlv::mData
```

Definition at line 315 of file include/openthread/netdiag.h

otLinkModeConfig

Represents an MLE Link Mode configuration.

Public Attributes

bool mRxOnWhenIdle
1, if the sender has its receiver on when not transmitting. 0, otherwise.

bool mDeviceType
1, if the sender is an FTD. 0, otherwise.

bool mNetworkData
1, if the sender requires the full Network Data. 0, otherwise.

Public Attribute Documentation

mRxOnWhenIdle

```
bool otLinkModeConfig::mRxOnWhenIdle
```

1, if the sender has its receiver on when not transmitting. 0, otherwise.

Definition at line 81 of file include/openthread/thread.h

mDeviceType

```
bool otLinkModeConfig::mDeviceType
```

1, if the sender is an FTD. 0, otherwise.

Definition at line 82 of file include/openthread/thread.h

mNetworkData

```
bool otLinkModeConfig::mNetworkData
```

1, if the sender requires the full Network Data. 0, otherwise.

Definition at line 83 of file include/openthread/thread.h

otNeighborInfo

Holds diagnostic information for a neighboring Thread node.

Public Attributes

otExtAddress	mExtAddress IEEE 802.15.4 Extended Address.
uint32_t	mAge Seconds since last heard.
uint32_t	mConnectionTime Seconds since link establishment (requires `CONFIG_UPTIME_ENABLE`)
uint16_t	mRloc16 RLOC16.
uint32_t	mLinkFrameCounter Link Frame Counter.
uint32_t	mMleFrameCounter MLE Frame Counter.
uint8_t	mLinkQualityIn Link Quality In.
int8_t	mAverageRssi Average RSSI.
int8_t	mLastRssi Last observed RSSI.
uint8_t	mLinkMargin Link Margin.
uint16_t	mFrameErrorRate Frame error rate (0xffff->100%). Requires error tracking feature.
uint16_t	mMessageErrorRate (IPv6) msg error rate (0xffff->100%). Requires error tracking feature.
uint16_t	mVersion Thread version of the neighbor.
bool	mRxOnWhenIdle rx-on-when-idle
bool	mFullThreadDevice Full Thread Device.
bool	mFullNetworkData Full Network Data.
bool	mIsChild Is the neighbor a child.

Public Attribute Documentation

mExtAddress

```
otExtAddress otNeighborInfo::mExtAddress
```

IEEE 802.15.4 Extended Address.

Definition at line `92` of file `include/openthread/thread.h`

mAge

```
uint32_t otNeighborInfo::mAge
```

Seconds since last heard.

Definition at line `93` of file `include/openthread/thread.h`

mConnectionTime

```
uint32_t otNeighborInfo::mConnectionTime
```

Seconds since link establishment (requires `CONFIG_UPTIME_ENABLE`)

Definition at line `94` of file `include/openthread/thread.h`

mRloc16

```
uint16_t otNeighborInfo::mRloc16
```

RLOC16.

Definition at line `95` of file `include/openthread/thread.h`

mLinkFrameCounter

```
uint32_t otNeighborInfo::mLinkFrameCounter
```

Link Frame Counter.

Definition at line `96` of file `include/openthread/thread.h`

mMleFrameCounter

```
uint32_t otNeighborInfo::mMleFrameCounter
```

MLE Frame Counter.

Definition at line `97` of file `include/openthread/thread.h`

mLinkQualityIn

```
uint8_t otNeighborInfo::mLinkQualityIn
```

Link Quality In.

Definition at line 98 of file include/openthread/thread.h

mAverageRssi

```
int8_t otNeighborInfo::mAverageRssi
```

Average RSSI.

Definition at line 99 of file include/openthread/thread.h

mLastRssi

```
int8_t otNeighborInfo::mLastRssi
```

Last observed RSSI.

Definition at line 100 of file include/openthread/thread.h

mLinkMargin

```
uint8_t otNeighborInfo::mLinkMargin
```

Link Margin.

Definition at line 101 of file include/openthread/thread.h

mFrameErrorRate

```
uint16_t otNeighborInfo::mFrameErrorRate
```

Frame error rate (0xffff->100%). Requires error tracking feature.

Definition at line 102 of file include/openthread/thread.h

mMessageErrorRate

```
uint16_t otNeighborInfo::mMessageErrorRate
```

(IPv6) msg error rate (0xffff->100%). Requires error tracking feature.

Definition at line 103 of file include/openthread/thread.h

mVersion

uint16_t otNeighborInfo::mVersion

Thread version of the neighbor.

Definition at line 104 of file include/openthread/thread.h

mRxOnWhenIdle

bool otNeighborInfo::mRxOnWhenIdle

rx-on-when-idle

Definition at line 105 of file include/openthread/thread.h

mFullThreadDevice

bool otNeighborInfo::mFullThreadDevice

Full Thread Device.

Definition at line 106 of file include/openthread/thread.h

mFullNetworkData

bool otNeighborInfo::mFullNetworkData

Full Network Data.

Definition at line 107 of file include/openthread/thread.h

mIsChild

bool otNeighborInfo::mIsChild

Is the neighbor a child.

Definition at line 108 of file include/openthread/thread.h

otLeaderData

Represents the Thread Leader Data.

Public Attributes

uint32_t	mPartitionId Partition ID.
uint8_t	mWeighting Leader Weight.
uint8_t	mDataVersion Full Network Data Version.
uint8_t	mStableDataVersion Stable Network Data Version.
uint8_t	mLeaderRouterId Leader Router ID.

Public Attribute Documentation

mPartitionId

```
uint32_t otLeaderData::mPartitionId
```

Partition ID.

Definition at line 121 of file include/openthread/thread.h

mWeighting

```
uint8_t otLeaderData::mWeighting
```

Leader Weight.

Definition at line 122 of file include/openthread/thread.h

mDataVersion

```
uint8_t otLeaderData::mDataVersion
```

Full Network Data Version.

Definition at line 123 of file include/openthread/thread.h

mStableDataVersion

```
uint8_t otLeaderData::mStableDataVersion
```

Stable Network Data Version.

Definition at line 124 of file include/openthread/thread.h

mLeaderRouterId

```
uint8_t otLeaderData::mLeaderRouterId
```

Leader Router ID.

Definition at line 125 of file include/openthread/thread.h

otRouterInfo

Holds diagnostic information for a Thread Router.

Public Attributes

otExtAddress	mExtAddress IEEE 802.15.4 Extended Address.
uint16_t	mRloc16 RLOC16.
uint8_t	mRouterId Router ID.
uint8_t	mNextHop Next hop to router.
uint8_t	mPathCost Path cost to router.
uint8_t	mLinkQualityIn Link Quality In.
uint8_t	mLinkQualityOut Link Quality Out.
uint8_t	mAge Time last heard.
bool	mAllocated Router ID allocated or not.
bool	mLinkEstablished Link established with Router ID or not.
uint8_t	mVersion Thread version.
uint8_t	mCslClockAccuracy Parent CSL parameters are only relevant when OPENTHREAD_CONFIG_MAC_CSL_RECEIVER_ENABLE is enabled.
uint8_t	mCslUncertainty CSL uncertainty, in ±10 us.

Public Attribute Documentation

mExtAddress

```
otExtAddress otRouterInfo::mExtAddress
```

IEEE 802.15.4 Extended Address.

Definition at line 134 of file include/openthread/thread.h

mRloc16

```
uint16_t otRouterInfo::mRloc16
```

RLOC16.

Definition at line 135 of file include/openthread/thread.h

mRouterId

```
uint8_t otRouterInfo::mRouterId
```

Router ID.

Definition at line 136 of file include/openthread/thread.h

mNextHop

```
uint8_t otRouterInfo::mNextHop
```

Next hop to router.

Definition at line 137 of file include/openthread/thread.h

mPathCost

```
uint8_t otRouterInfo::mPathCost
```

Path cost to router.

Definition at line 138 of file include/openthread/thread.h

mLinkQualityIn

```
uint8_t otRouterInfo::mLinkQualityIn
```

Link Quality In.

Definition at line 139 of file include/openthread/thread.h

mLinkQualityOut

```
uint8_t otRouterInfo::mLinkQualityOut
```

Link Quality Out.

Definition at line 140 of file include/openthread/thread.h

mAge

```
uint8_t otRouterInfo::mAge
```

Time last heard.

Definition at line 141 of file include/openthread/thread.h

mAllocated

```
bool otRouterInfo::mAllocated
```

Router ID allocated or not.

Definition at line 142 of file include/openthread/thread.h

mLinkEstablished

```
bool otRouterInfo::mLinkEstablished
```

Link established with Router ID or not.

Definition at line 143 of file include/openthread/thread.h

mVersion

```
uint8_t otRouterInfo::mVersion
```

Thread version.

Definition at line 144 of file include/openthread/thread.h

mCslClockAccuracy

```
uint8_t otRouterInfo::mCslClockAccuracy
```

Parent CSL parameters are only relevant when OPENTHREAD_CONFIG_MAC_CSL_RECEIVER_ENABLE is enabled.

CSL clock accuracy, in ± ppm

Definition at line 150 of file include/openthread/thread.h

mCslUncertainty

```
uint8_t otRouterInfo::mCslUncertainty
```

CSL uncertainty, in ±10 us.

Definition at line 151 of file include/openthread/thread.h

otIpCounters

Represents the IP level counters.

Public Attributes

uint32_t	mTxSuccess The number of IPv6 packets successfully transmitted.
uint32_t	mRxSuccess The number of IPv6 packets successfully received.
uint32_t	mTxFailure The number of IPv6 packets failed to transmit.
uint32_t	mRxFailure The number of IPv6 packets failed to receive.

Public Attribute Documentation

mTxSuccess

```
uint32_t otIpCounters::mTxSuccess
```

The number of IPv6 packets successfully transmitted.

Definition at line `160` of file `include/openthread/thread.h`

mRxSuccess

```
uint32_t otIpCounters::mRxSuccess
```

The number of IPv6 packets successfully received.

Definition at line `161` of file `include/openthread/thread.h`

mTxFailure

```
uint32_t otIpCounters::mTxFailure
```

The number of IPv6 packets failed to transmit.

Definition at line `162` of file `include/openthread/thread.h`

mRxFailure

```
uint32_t otIpCounters::mRxFailure
```

The number of IPv6 packets failed to receive.

Definition at line 163 of file include/openthread/thread.h

otMleCounters

Represents the Thread MLE counters.

Public Attributes

uint16_t	mDisabledRole Number of times device entered OT_DEVICE_ROLE_DISABLED role.
uint16_t	mDetachedRole Number of times device entered OT_DEVICE_ROLE_DETACHED role.
uint16_t	mChildRole Number of times device entered OT_DEVICE_ROLE_CHILD role.
uint16_t	mRouterRole Number of times device entered OT_DEVICE_ROLE_ROUTER role.
uint16_t	mLeaderRole Number of times device entered OT_DEVICE_ROLE_LEADER role.
uint16_t	mAttachAttempts Number of attach attempts while device was detached.
uint16_t	mPartitionIdChanges Number of changes to partition ID.
uint16_t	mBetterPartitionAttachAttempts Number of attempts to attach to a better partition.
uint64_t	mDisabledTime Role time tracking.
uint64_t	mDetachedTime Number of milliseconds device has been in OT_DEVICE_ROLE_DETACHED role.
uint64_t	mChildTime Number of milliseconds device has been in OT_DEVICE_ROLE_CHILD role.
uint64_t	mRouterTime Number of milliseconds device has been in OT_DEVICE_ROLE_ROUTER role.
uint64_t	mLeaderTime Number of milliseconds device has been in OT_DEVICE_ROLE_LEADER role.
uint64_t	mTrackedTime Number of milliseconds tracked by previous counters.
uint16_t	mParentChanges Number of times device changed its parent.

Public Attribute Documentation

mDisabledRole

uint16_t otMleCounters::mDisabledRole

Number of times device entered OT_DEVICE_ROLE_DISABLED role.

Definition at line 172 of file include/openthread/thread.h

mDetachedRole

uint16_t otMleCounters::mDetachedRole

Number of times device entered OT_DEVICE_ROLE_DETACHED role.

Definition at line 173 of file include/openthread/thread.h

mChildRole

uint16_t otMleCounters::mChildRole

Number of times device entered OT_DEVICE_ROLE_CHILD role.

Definition at line 174 of file include/openthread/thread.h

mRouterRole

uint16_t otMleCounters::mRouterRole

Number of times device entered OT_DEVICE_ROLE_ROUTER role.

Definition at line 175 of file include/openthread/thread.h

mLeaderRole

uint16_t otMleCounters::mLeaderRole

Number of times device entered OT_DEVICE_ROLE_LEADER role.

Definition at line 176 of file include/openthread/thread.h

mAttachAttempts

uint16_t otMleCounters::mAttachAttempts

Number of attach attempts while device was detached.

Definition at line 177 of file include/openthread/thread.h

mPartitionIdChanges

uint16_t otMleCounters::mPartitionIdChanges

Number of changes to partition ID.

Definition at line 178 of file include/openthread/thread.h

mBetterPartitionAttachAttempts

uint16_t otMleCounters::mBetterPartitionAttachAttempts

Number of attempts to attach to a better partition.

Definition at line 179 of file include/openthread/thread.h

mDisabledTime

uint64_t otMleCounters::mDisabledTime

Role time tracking.

When uptime feature is enabled (OPENTHREAD_CONFIG_UPTIME_ENABLE = 1) time spent in each MLE role is tracked. Number of milliseconds device has been in OT_DEVICE_ROLE_DISABLED role.

Definition at line 187 of file include/openthread/thread.h

mDetachedTime

uint64_t otMleCounters::mDetachedTime

Number of milliseconds device has been in OT_DEVICE_ROLE_DETACHED role.

Definition at line 188 of file include/openthread/thread.h

mChildTime

uint64_t otMleCounters::mChildTime

Number of milliseconds device has been in OT_DEVICE_ROLE_CHILD role.

Definition at line 189 of file include/openthread/thread.h

mRouterTime

uint64_t otMleCounters::mRouterTime

Number of milliseconds device has been in OT_DEVICE_ROLE_ROUTER role.

Definition at line 190 of file include/openthread/thread.h

uint64_t otMleCounters::mLeaderTime

Number of milliseconds device has been in OT_DEVICE_ROLE_LEADER role.

Definition at line 191 of file include/openthread/thread.h

mTrackedTime

uint64_t otMleCounters::mTrackedTime

Number of milliseconds tracked by previous counters.

Definition at line 192 of file include/openthread/thread.h

mParentChanges

uint16_t otMleCounters::mParentChanges

Number of times device changed its parent.

A parent change can happen if device detaches from its current parent and attaches to a different one, or even while device is attached when the periodic parent search feature is enabled (please see option OPENTHREAD_CONFIG_PARENT_SEARCH_ENABLE).

Definition at line 202 of file include/openthread/thread.h

otThreadParentResponseInfo

Represents the MLE Parent Response data.

Public Attributes

otExtAddress	mExtAddr IEEE 802.15.4 Extended Address of the Parent.
uint16_t	mRloc16 Short address of the Parent.
int8_t	mRssi Rssi of the Parent.
int8_t	mPriority Parent priority.
uint8_t	mLinkQuality3 Parent Link Quality 3.
uint8_t	mLinkQuality2 Parent Link Quality 2.
uint8_t	mLinkQuality1 Parent Link Quality 1.
bool	mIsAttached Is the node receiving parent response attached.

Public Attribute Documentation

mExtAddr

```
otExtAddress otThreadParentResponseInfo::mExtAddr
```

IEEE 802.15.4 Extended Address of the Parent.

Definition at line 211 of file include/openthread/thread.h

mRloc16

```
uint16_t otThreadParentResponseInfo::mRloc16
```

Short address of the Parent.

Definition at line 212 of file include/openthread/thread.h

mRssi

```
int8_t otThreadParentResponseInfo::mRssi
```

Rssi of the Parent.

Definition at line 213 of file include/openthread/thread.h

mPriority

```
int8_t otThreadParentResponseInfo::mPriority
```

Parent priority.

Definition at line 214 of file include/openthread/thread.h

mLinkQuality3

```
uint8_t otThreadParentResponseInfo::mLinkQuality3
```

Parent Link Quality 3.

Definition at line 215 of file include/openthread/thread.h

mLinkQuality2

```
uint8_t otThreadParentResponseInfo::mLinkQuality2
```

Parent Link Quality 2.

Definition at line 216 of file include/openthread/thread.h

mLinkQuality1

```
uint8_t otThreadParentResponseInfo::mLinkQuality1
```

Parent Link Quality 1.

Definition at line 217 of file include/openthread/thread.h

mIsAttached

```
bool otThreadParentResponseInfo::mIsAttached
```

Is the node receiving parent response attached.

Definition at line 218 of file include/openthread/thread.h

otThreadDiscoveryRequestInfo

Represents the Thread Discovery Request data.

Public Attributes

otExtAddress	mExtAddress IEEE 802.15.4 Extended Address of the requester.
uint8_t	mVersion Thread version.
bool	mIsJoiner Whether is from joiner.

Public Attribute Documentation

mExtAddress

```
otExtAddress otThreadDiscoveryRequestInfo::mExtAddress
```

IEEE 802.15.4 Extended Address of the requester.

Definition at line 1011 of file include/openthread/thread.h

mVersion

```
uint8_t otThreadDiscoveryRequestInfo::mVersion
```

Thread version.

Definition at line 1012 of file include/openthread/thread.h

mIsJoiner

```
bool otThreadDiscoveryRequestInfo::mIsJoiner
```

Whether is from joiner.

Definition at line 1013 of file include/openthread/thread.h

Joiner

Joiner

This module includes functions for the Thread Joiner role.

Note

- The functions in this module require `OPENTHREAD_CONFIG_JOINER_ENABLE=1`.

Modules

otJoinerDiscerner

Enumerations

enum otJoinerState {

```
OT_JOINER_STATE_IDLE = 0
OT_JOINER_STATE_DISCOVER = 1
OT_JOINER_STATE_CONNECT = 2
OT_JOINER_STATE_CONNECTED = 3
OT_JOINER_STATE_ENTRUST = 4
OT_JOINER_STATE_JOINED = 5
```

}

Defines the Joiner State.

Typedefs

typedef enum otJoinerState — otJoinerState
Defines the Joiner State.

typedef struct otJoinerDiscerner — otJoinerDiscerner
Represents a Joiner Discerner.

typedef void(* — otJoinerCallback)(otError aError, void *aContext)
Pointer is called to notify the completion of a join operation.

Functions

otError — otJoinerStart(otInstance *aInstance, const char *aPskd, const char *aProvisioningUrl, const char *aVendorName, const char *aVendorModel, const char *aVendorSwVersion, const char *aVendorData, otJoinerCallback aCallback, void *aContext)
Enables the Thread Joiner role.

void — otJoinerStop(otInstance *aInstance)
Disables the Thread Joiner role.

otJoinerState — otJoinerGetState(otInstance *aInstance)
Gets the Joiner State.

const otExtAddress * — otJoinerGetId(otInstance *aInstance)
Gets the Joiner ID.

otError	otJoinerSetDiscerner(otInstance *aInstance, otJoinerDiscerner *aDiscerner) Sets the Joiner Discerner.
const otJoinerDiscerner *	otJoinerGetDiscerner(otInstance *aInstance) Gets the Joiner Discerner.
const char *	otJoinerStateToString(otJoinerState aState) Converts a given joiner state enumeration value to a human-readable string.

Macros

#define	OT_JOINER_MAX_DISCERNER_LENGTH 64 Maximum length of a Joiner Discerner in bits.

Enumeration Documentation

otJoinerState

```
otJoinerState
```

Defines the Joiner State.

Enumerator	
OT_JOINER_STATE_IDLE	
OT_JOINER_STATE_DISCOVER	
OT_JOINER_STATE_CONNECT	
OT_JOINER_STATE_CONNECTED	
OT_JOINER_STATE_ENTRUST	
OT_JOINER_STATE_JOINED	

Definition at line 62 of file include/openthread/joiner.h

Typedef Documentation

otJoinerState

```
typedef enum otJoinerState otJoinerState
```

Defines the Joiner State.

Definition at line 70 of file include/openthread/joiner.h

otJoinerDiscerner

```
typedef struct otJoinerDiscerner otJoinerDiscerner
```

Represents a Joiner Discerner.

Definition at line 82 of file include/openthread/joiner.h

otJoinerCallback

```
typedef void(* otJoinerCallback) (otError aError, void *aContext) )(otError aError, void *aContext)
```

Pointer is called to notify the completion of a join operation.

Parameters

[in]	aError	OT_ERROR_NONE if the join process succeeded. OT_ERROR_SECURITY if the join process failed due to security credentials. OT_ERROR_NOT_FOUND if no joinable network was discovered. OT_ERROR_RESPONSE_TIMEOUT if a response timed out.
[in]	aContext	A pointer to application-specific context.

Definition at line `94` of file `include/openthread/joiner.h`

Function Documentation

otJoinerStart

```
otError otJoinerStart (otInstance *aInstance, const char *aPskd, const char *aProvisioningUrl, const char *aVendorName, const char *aVendorModel, const char *aVendorSwVersion, const char *aVendorData, otJoinerCallback aCallback, void *aContext)
```

Enables the Thread Joiner role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPskd	A pointer to the PSKd.
[in]	aProvisioningUrl	A pointer to the Provisioning URL (may be NULL).
[in]	aVendorName	A pointer to the Vendor Name (may be NULL).
[in]	aVendorModel	A pointer to the Vendor Model (may be NULL).
[in]	aVendorSwVersion	A pointer to the Vendor SW Version (may be NULL).
[in]	aVendorData	A pointer to the Vendor Data (may be NULL).
[in]	aCallback	A pointer to a function that is called when the join operation completes.
[in]	aContext	A pointer to application-specific context.

Definition at line `115` of file `include/openthread/joiner.h`

otJoinerStop

```
void otJoinerStop (otInstance *aInstance)
```

Disables the Thread Joiner role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `131` of file `include/openthread/joiner.h`

otJoinerGetState

> otJoinerState otJoinerGetState (otInstance *aInstance)

Gets the Joiner State.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The joiner state.

Definition at line `141` of file `include/openthread/joiner.h`

otJoinerGetId

> const otExtAddress * otJoinerGetId (otInstance *aInstance)

Gets the Joiner ID.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

If a Joiner Discerner is not set, Joiner ID is the first 64 bits of the result of computing SHA-256 over factory-assigned IEEE EUI-64. Otherwise the Joiner ID is calculated from the Joiner Discerner value.

The Joiner ID is also used as the device's IEEE 802.15.4 Extended Address during the commissioning process.

Returns

- A pointer to the Joiner ID.

Definition at line `156` of file `include/openthread/joiner.h`

otJoinerSetDiscerner

> otError otJoinerSetDiscerner (otInstance *aInstance, otJoinerDiscerner *aDiscerner)

Sets the Joiner Discerner.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aDiscerner	A pointer to a Joiner Discerner. If NULL clears any previously set discerner.

The Joiner Discerner is used to calculate the Joiner ID during the Thread Commissioning process. For more information, refer to otJoinerGetId. **Note**

- The Joiner Discerner takes the place of the Joiner EUI-64 during the joiner session of Thread Commissioning.

Definition at line `173` of file `include/openthread/joiner.h`

otJoinerGetDiscerner

> const otJoinerDiscerner * otJoinerGetDiscerner (otInstance *aInstance)

Gets the Joiner Discerner.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

For more information, refer to otJoinerSetDiscerner.

Returns

- A pointer to Joiner Discerner or NULL if none is set.

Definition at line `183` of file `include/openthread/joiner.h`

otJoinerStateToString

```
const char * otJoinerStateToString (otJoinerState aState)
```

Converts a given joiner state enumeration value to a human-readable string.

Parameters

[in]	aState	The joiner state.

Returns

- A human-readable string representation of `aState`.

Definition at line `193` of file `include/openthread/joiner.h`

Macro Definition Documentation

OT_JOINER_MAX_DISCERNER_LENGTH

```
#define OT_JOINER_MAX_DISCERNER_LENGTH
```

Value:

```
64
```

Maximum length of a Joiner Discerner in bits.

Definition at line `72` of file `include/openthread/joiner.h`

otJoinerDiscerner

Represents a Joiner Discerner.

Public Attributes

uint64_t	mValue Discerner value (the lowest `mLength` bits specify the discerner).
uint8_t	mLength Length (number of bits) - must be non-zero and at most `OT_JOINER_MAX_DISCERNER_LENGTH`.

Public Attribute Documentation

mValue

```
uint64_t otJoinerDiscerner::mValue
```

Discerner value (the lowest `mLength` bits specify the discerner).

Definition at line `80` of file `include/openthread/joiner.h`

mLength

```
uint8_t otJoinerDiscerner::mLength
```

Length (number of bits) - must be non-zero and at most `OT_JOINER_MAX_DISCERNER_LENGTH`.

Definition at line `81` of file `include/openthread/joiner.h`

Operational Dataset

Operational Dataset

Includes functions for the Operational Dataset API.

For FTD builds only, Dataset Updater includes functions to manage dataset updates.

For FTD and MTD builds, the Operational Dataset API includes functions to manage Active and Pending datasets and dataset TLVs.

Modules

otNetworkKey

otNetworkName

otExtendedPanId

otPskc

otSecurityPolicy

otOperationalDatasetComponents

otTimestamp

otOperationalDataset

otOperationalDatasetTlvs

Enumerations

enum otMeshcopTlvType {

```
OT_MESHCOP_TLV_CHANNEL = 0
OT_MESHCOP_TLV_PANID = 1
OT_MESHCOP_TLV_EXTPANID = 2
OT_MESHCOP_TLV_NETWORKNAME = 3
OT_MESHCOP_TLV_PSKC = 4
OT_MESHCOP_TLV_NETWORKKEY = 5
OT_MESHCOP_TLV_NETWORK_KEY_SEQUENCE = 6
OT_MESHCOP_TLV_MESHLOCALPREFIX = 7
OT_MESHCOP_TLV_STEERING_DATA = 8
OT_MESHCOP_TLV_BORDER_AGENT_RLOC = 9
OT_MESHCOP_TLV_COMMISSIONER_ID = 10
OT_MESHCOP_TLV_COMM_SESSION_ID = 11
OT_MESHCOP_TLV_SECURITYPOLICY = 12
OT_MESHCOP_TLV_GET = 13
OT_MESHCOP_TLV_ACTIVETIMESTAMP = 14
OT_MESHCOP_TLV_COMMISSIONER_UDP_PORT = 15
OT_MESHCOP_TLV_STATE = 16
OT_MESHCOP_TLV_JOINER_DTLS = 17
OT_MESHCOP_TLV_JOINER_UDP_PORT = 18
OT_MESHCOP_TLV_JOINER_IID = 19
OT_MESHCOP_TLV_JOINER_RLOC = 20
OT_MESHCOP_TLV_JOINER_ROUTER_KEK = 21
OT_MESHCOP_TLV_PROVISIONING_URL = 32
OT_MESHCOP_TLV_VENDOR_NAME_TLV = 33
OT_MESHCOP_TLV_VENDOR_MODEL_TLV = 34
OT_MESHCOP_TLV_VENDOR_SW_VERSION_TLV = 35
OT_MESHCOP_TLV_VENDOR_DATA_TLV = 36
OT_MESHCOP_TLV_VENDOR_STACK_VERSION_TLV = 37
OT_MESHCOP_TLV_UDP_ENCAPSULATION_TLV = 48
OT_MESHCOP_TLV_IPV6_ADDRESS_TLV = 49
OT_MESHCOP_TLV_PENDINGTIMESTAMP = 51
OT_MESHCOP_TLV_DELAYTIMER = 52
OT_MESHCOP_TLV_CHANNELMASK = 53
OT_MESHCOP_TLV_COUNT = 54
OT_MESHCOP_TLV_PERIOD = 55
OT_MESHCOP_TLV_SCAN_DURATION = 56
OT_MESHCOP_TLV_ENERGY_LIST = 57
OT_MESHCOP_TLV_DISCOVERYREQUEST = 128
OT_MESHCOP_TLV_DISCOVERYRESPONSE = 129
OT_MESHCOP_TLV_JOINERADVERTISEMENT = 241
```

}

Represents meshcop TLV types.

Typedefs

typedef struct otNetworkKey otNetworkKey

Represents a Thread Network Key.

typedef otCryptoKeyRef otNetworkKeyRef

This datatype represents KeyRef to NetworkKey.

typedef struct otNetworkName otNetworkName

Represents a Network Name.

typedef struct otExtendedPanId otExtendedPanId

Represents an Extended PAN ID.

typedef otIp6NetworkPrefix otMeshLocalPrefix

Represents a Mesh Local Prefix.

typedef struct otPskc otPskc

Represents a PSKc.

typedef otCryptoKeyRef — otPskcRef
This datatype represents KeyRef to PSKc.

typedef struct otSecurityPolicy — otSecurityPolicy
Represent Security Policy.

typedef uint32_t — otChannelMask
Represents Channel Mask.

typedef struct otOperationalDatasetComponents — otOperationalDatasetComponents
Represents presence of different components in Active or Pending Operational Dataset.

typedef struct otTimestamp — otTimestamp
Represents a Thread Dataset timestamp component.

typedef struct otOperationalDataset — otOperationalDataset
Represents an Active or Pending Operational Dataset.

typedef struct otOperationalDatasetTlvs — otOperationalDatasetTlvs
Represents an Active or Pending Operational Dataset.

typedef enum otMeshcopTlvType — otMeshcopTlvType
Represents meshcop TLV types.

typedef void(* — otDatasetMgmtSetCallback)(otError aResult, void *aContext)
Pointer is called when a response to a MGMT_SET request is received or times out.

typedef void(* — otDatasetUpdaterCallback)(otError aError, void *aContext)
This callback function pointer is called when a Dataset update request finishes, reporting success or failure status of the Dataset update request.

Variables

OT_TOOL_PACKED_BEGIN struct otNetworkKey — OT_TOOL_PACKED_END

Functions

bool — otDatasetIsCommissioned(otInstance *aInstance)
Indicates whether a valid network is present in the Active Operational Dataset or not.

otError — otDatasetGetActive(otInstance *aInstance, otOperationalDataset *aDataset)
Gets the Active Operational Dataset.

otError — otDatasetGetActiveTlvs(otInstance *aInstance, otOperationalDatasetTlvs *aDataset)
Gets the Active Operational Dataset.

otError — otDatasetSetActive(otInstance *aInstance, const otOperationalDataset *aDataset)
Sets the Active Operational Dataset.

otError — otDatasetSetActiveTlvs(otInstance *aInstance, const otOperationalDatasetTlvs *aDataset)
Sets the Active Operational Dataset.

otError — otDatasetGetPending(otInstance *aInstance, otOperationalDataset *aDataset)
Gets the Pending Operational Dataset.

otError otDatasetGetPendingTlvs(otInstance *aInstance, otOperationalDatasetTlvs *aDataset)
Gets the Pending Operational Dataset.

otError otDatasetSetPending(otInstance *aInstance, const otOperationalDataset *aDataset)
Sets the Pending Operational Dataset.

otError otDatasetSetPendingTlvs(otInstance *aInstance, const otOperationalDatasetTlvs *aDataset)
Sets the Pending Operational Dataset.

otError otDatasetSendMgmtActiveGet(otInstance *aInstance, const otOperationalDatasetComponents *aDatasetComponents, const uint8_t *aTlvTypes, uint8_t aLength, const otIp6Address *aAddress)
Sends MGMT_ACTIVE_GET.

otError otDatasetSendMgmtActiveSet(otInstance *aInstance, const otOperationalDataset *aDataset, const uint8_t *aTlvs, uint8_t aLength, otDatasetMgmtSetCallback aCallback, void *aContext)
Sends MGMT_ACTIVE_SET.

otError otDatasetSendMgmtPendingGet(otInstance *aInstance, const otOperationalDatasetComponents *aDatasetComponents, const uint8_t *aTlvTypes, uint8_t aLength, const otIp6Address *aAddress)
Sends MGMT_PENDING_GET.

otError otDatasetSendMgmtPendingSet(otInstance *aInstance, const otOperationalDataset *aDataset, const uint8_t *aTlvs, uint8_t aLength, otDatasetMgmtSetCallback aCallback, void *aContext)
Sends MGMT_PENDING_SET.

otError otDatasetGeneratePskc(const char *aPassPhrase, const otNetworkName *aNetworkName, const otExtendedPanId *aExtPanId, otPskc *aPskc)
Generates PSKc from a given pass-phrase, network name, and extended PAN ID.

otError otNetworkNameFromString(otNetworkName *aNetworkName, const char *aNameString)
Sets an `otNetworkName` instance from a given null terminated C string.

otError otDatasetParseTlvs(const otOperationalDatasetTlvs *aDatasetTlvs, otOperationalDataset *aDataset)
Parses an Operational Dataset from a given `otOperationalDatasetTlvs`.

otError otDatasetConvertToTlvs(const otOperationalDataset *aDataset, otOperationalDatasetTlvs *aDatasetTlvs)
Converts a given Operational Dataset to `otOperationalDatasetTlvs`.

otError otDatasetUpdateTlvs(const otOperationalDataset *aDataset, otOperationalDatasetTlvs *aDatasetTlvs)
Updates a given Operational Dataset.

otError otDatasetCreateNewNetwork(otInstance *aInstance, otOperationalDataset *aDataset)
For FTD only, creates a new Operational Dataset to use when forming a new network.

uint32_t otDatasetGetDelayTimerMinimal(otInstance *aInstance)
For FTD only, gets a minimal delay timer.

otError otDatasetSetDelayTimerMinimal(otInstance *aInstance, uint32_t aDelayTimerMinimal)
For FTD only, sets a minimal delay timer.

otError otDatasetUpdaterRequestUpdate(otInstance *aInstance, const otOperationalDataset *aDataset, otDatasetUpdaterCallback aCallback, void *aContext)
Requests an update to Operational Dataset.

void otDatasetUpdaterCancelUpdate(otInstance *aInstance)
Cancels an ongoing (if any) Operational Dataset update request.

bool otDatasetUpdaterIsUpdateOngoing(otInstance *aInstance)
Indicates whether there is an ongoing Operation Dataset update request.

Macros

#define OT_NETWORK_KEY_SIZE 16
Size of the Thread Network Key (bytes)

#define OT_NETWORK_NAME_MAX_SIZE 16
Maximum size of the Thread Network Name field (bytes)

#define OT_EXT_PAN_ID_SIZE 8
Size of a Thread PAN ID (bytes)

#define OT_MESH_LOCAL_PREFIX_SIZE OT_IP6_PREFIX_SIZE
Size of the Mesh Local Prefix (bytes)

#define OT_PSKC_MAX_SIZE 16
Maximum size of the PSKc (bytes)

#define OT_CHANNEL_1_MASK (1 << 1)
Channel 1.

#define OT_CHANNEL_2_MASK (1 << 2)
Channel 2.

#define OT_CHANNEL_3_MASK (1 << 3)
Channel 3.

#define OT_CHANNEL_4_MASK (1 << 4)
Channel 4.

#define OT_CHANNEL_5_MASK (1 << 5)
Channel 5.

#define OT_CHANNEL_6_MASK (1 << 6)
Channel 6.

#define OT_CHANNEL_7_MASK (1 << 7)
Channel 7.

#define OT_CHANNEL_8_MASK (1 << 8)
Channel 8.

#define OT_CHANNEL_9_MASK (1 << 9)
Channel 9.

#define OT_CHANNEL_10_MASK (1 << 10)
Channel 10.

#define OT_CHANNEL_11_MASK (1 << 11)
Channel 11.

#define OT_CHANNEL_12_MASK (1 << 12)
Channel 12.

#define OT_CHANNEL_13_MASK (1 << 13)
Channel 13.

#define OT_CHANNEL_14_MASK (1 << 14)
Channel 14.

#define OT_CHANNEL_15_MASK (1 << 15)
Channel 15.

#define OT_CHANNEL_16_MASK (1 << 16)
Channel 16.

#define OT_CHANNEL_17_MASK (1 << 17)
Channel 17.

#define OT_CHANNEL_18_MASK (1 << 18)
Channel 18.

#define OT_CHANNEL_19_MASK (1 << 19)
Channel 19.

#define OT_CHANNEL_20_MASK (1 << 20)
Channel 20.

#define OT_CHANNEL_21_MASK (1 << 21)
Channel 21.

#define OT_CHANNEL_22_MASK (1 << 22)
Channel 22.

#define OT_CHANNEL_23_MASK (1 << 23)
Channel 23.

#define OT_CHANNEL_24_MASK (1 << 24)
Channel 24.

#define OT_CHANNEL_25_MASK (1 << 25)
Channel 25.

#define OT_CHANNEL_26_MASK (1 << 26)
Channel 26.

#define OT_OPERATIONAL_DATASET_MAX_LENGTH 254
Maximum length of Operational Dataset in bytes.

Enumeration Documentation

otMeshcopTlvType

otMeshcopTlvType

Represents meshcop TLV types.

Enumerator	
OT_MESHCOP_TLV_CHANNEL	meshcop Channel TLV
OT_MESHCOP_TLV_PANID	meshcop Pan Id TLV
OT_MESHCOP_TLV_EXTPANID	meshcop Extended Pan Id TLV
OT_MESHCOP_TLV_NETWORKNAME	meshcop Network Name TLV
OT_MESHCOP_TLV_PSKC	meshcop PSKc TLV
OT_MESHCOP_TLV_NETWORKKEY	meshcop Network Key TLV
OT_MESHCOP_TLV_NETWORK_KEY_SEQUENCE	meshcop Network Key Sequence TLV
OT_MESHCOP_TLV_MESHLOCALPREFIX	meshcop Mesh Local Prefix TLV
OT_MESHCOP_TLV_STEERING_DATA	meshcop Steering Data TLV
OT_MESHCOP_TLV_BORDER_AGENT_RLOC	meshcop Border Agent Locator TLV
OT_MESHCOP_TLV_COMMISSIONER_ID	meshcop Commissioner ID TLV
OT_MESHCOP_TLV_COMM_SESSION_ID	meshcop Commissioner Session ID TLV
OT_MESHCOP_TLV_SECURITYPOLICY	meshcop Security Policy TLV
OT_MESHCOP_TLV_GET	meshcop Get TLV
OT_MESHCOP_TLV_ACTIVETIMESTAMP	meshcop Active Timestamp TLV

OT_MESHCOP_TLV_COMMISSIONER_UDP_PORT	meshcop Commissioner UDP Port TLV
OT_MESHCOP_TLV_STATE	meshcop State TLV
OT_MESHCOP_TLV_JOINER_DTLS	meshcop Joiner DTLS Encapsulation TLV
OT_MESHCOP_TLV_JOINER_UDP_PORT	meshcop Joiner UDP Port TLV
OT_MESHCOP_TLV_JOINER_IID	meshcop Joiner IID TLV
OT_MESHCOP_TLV_JOINER_RLOC	meshcop Joiner Router Locator TLV
OT_MESHCOP_TLV_JOINER_ROUTER_KEK	meshcop Joiner Router KEK TLV
OT_MESHCOP_TLV_PROVISIONING_URL	meshcop Provisioning URL TLV
OT_MESHCOP_TLV_VENDOR_NAME_TLV	meshcop Vendor Name TLV
OT_MESHCOP_TLV_VENDOR_MODEL_TLV	meshcop Vendor Model TLV
OT_MESHCOP_TLV_VENDOR_SW_VERSION_TLV	meshcop Vendor SW Version TLV
OT_MESHCOP_TLV_VENDOR_DATA_TLV	meshcop Vendor Data TLV
OT_MESHCOP_TLV_VENDOR_STACK_VERSION_TLV	meshcop Vendor Stack Version TLV
OT_MESHCOP_TLV_UDP_ENCAPSULATION_TLV	meshcop UDP encapsulation TLV
OT_MESHCOP_TLV_IPV6_ADDRESS_TLV	meshcop IPv6 address TLV
OT_MESHCOP_TLV_PENDINGTIMESTAMP	meshcop Pending Timestamp TLV
OT_MESHCOP_TLV_DELAYTIMER	meshcop Delay Timer TLV
OT_MESHCOP_TLV_CHANNELMASK	meshcop Channel Mask TLV
OT_MESHCOP_TLV_COUNT	meshcop Count TLV
OT_MESHCOP_TLV_PERIOD	meshcop Period TLV
OT_MESHCOP_TLV_SCAN_DURATION	meshcop Scan Duration TLV
OT_MESHCOP_TLV_ENERGY_LIST	meshcop Energy List TLV
OT_MESHCOP_TLV_DISCOVERYREQUEST	meshcop Discovery Request TLV
OT_MESHCOP_TLV_DISCOVERYRESPONSE	meshcop Discovery Response TLV
OT_MESHCOP_TLV_JOINERADVERTISEMENT	meshcop Joiner Advertisement TLV

Definition at line `274` of file `include/openthread/dataset.h`

Typedef Documentation

otNetworkKey

```
typedef struct otNetworkKey otNetworkKey
```

Represents a Thread Network Key.

Definition at line `74` of file `include/openthread/dataset.h`

otNetworkKeyRef

```
typedef otCryptoKeyRef otNetworkKeyRef
```

This datatype represents KeyRef to NetworkKey.

Reference to Key

Definition at line 80 of file include/openthread/dataset.h

otNetworkName

```
typedef struct otNetworkName otNetworkName
```

Represents a Network Name.

The otNetworkName is a null terminated C string (i.e., m8 char array MUST end with null char \0).

Definition at line 93 of file include/openthread/dataset.h

otExtendedPanId

```
typedef struct otExtendedPanId otExtendedPanId
```

Represents an Extended PAN ID.

Definition at line 111 of file include/openthread/dataset.h

otMeshLocalPrefix

```
typedef otIp6NetworkPrefix otMeshLocalPrefix
```

Represents a Mesh Local Prefix.

Definition at line 119 of file include/openthread/dataset.h

otPskc

```
typedef struct otPskc otPskc
```

Represents a PSKc.

Definition at line 137 of file include/openthread/dataset.h

otPskcRef

```
typedef otCryptoKeyRef otPskcRef
```

This datatype represents KeyRef to PSKc.

Reference to Key

Definition at line 143 of file include/openthread/dataset.h

otSecurityPolicy

```
typedef struct otSecurityPolicy otSecurityPolicy
```

Represent Security Policy.

Definition at line 163 of file include/openthread/dataset.h

otChannelMask

```
typedef uint32_t otChannelMask
```

Represents Channel Mask.

Definition at line 169 of file include/openthread/dataset.h

otOperationalDatasetComponents

```
typedef struct otOperationalDatasetComponents otOperationalDatasetComponents
```

Represents presence of different components in Active or Pending Operational Dataset.

Definition at line 216 of file include/openthread/dataset.h

otTimestamp

```
typedef struct otTimestamp otTimestamp
```

Represents a Thread Dataset timestamp component.

Definition at line 227 of file include/openthread/dataset.h

otOperationalDataset

```
typedef struct otOperationalDataset otOperationalDataset
```

Represents an Active or Pending Operational Dataset.

Components in Dataset are optional. `mComponents` structure specifies which components are present in the Dataset.

Definition at line 250 of file include/openthread/dataset.h

otOperationalDatasetTlvs

```
typedef struct otOperationalDatasetTlvs otOperationalDatasetTlvs
```

Represents an Active or Pending Operational Dataset.

The Operational Dataset is TLV encoded as specified by Thread.

Definition at line 268 of file include/openthread/dataset.h

otMeshcopTlvType

typedef enum otMeshcopTlvType otMeshcopTlvType

Represents meshcop TLV types.

Definition at line 316 of file include/openthread/dataset.h

otDatasetMgmtSetCallback

typedef void(* otDatasetMgmtSetCallback) (otError aResult, void *aContext))(otError aResult, void *aContext)

Pointer is called when a response to a MGMT_SET request is received or times out.

Parameters

[in]	aResult	A result of the operation.
[in]	aContext	A pointer to application-specific context.

Definition at line 331 of file include/openthread/dataset.h

otDatasetUpdaterCallback

typedef void(* otDatasetUpdaterCallback) (otError aError, void *aContext))(otError aError, void *aContext)

This callback function pointer is called when a Dataset update request finishes, reporting success or failure status of the Dataset update request.

Parameters

[in]	aError	The error status. OT_ERROR_NONE indicates successful Dataset update. OT_ERROR_INVALID_STATE indicates failure due invalid state (MLE being disabled). OT_ERROR_ALREADY indicates failure due to another device within network requesting a conflicting Dataset update.
[in]	aContext	A pointer to the arbitrary context (provided by user in otDatasetUpdaterRequestUpdate()).

Available when OPENTHREAD_CONFIG_DATASET_UPDATER_ENABLE is enabled.

Definition at line 69 of file include/openthread/dataset_updater.h

Variable Documentation

OT_TOOL_PACKED_END

OT_TOOL_PACKED_BEGIN struct otPskc OT_TOOL_PACKED_END

Definition at line 68 of file include/openthread/dataset.h

Function Documentation

otDatasetIsCommissioned

bool otDatasetIsCommissioned (otInstance *aInstance)

Indicates whether a valid network is present in the Active Operational Dataset or not.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- TRUE if a valid network is present in the Active Operational Dataset, FALSE otherwise.

Definition at line `341` of file `include/openthread/dataset.h`

otDatasetGetActive

```
otError otDatasetGetActive (otInstance *aInstance, otOperationalDataset *aDataset)
```

Gets the Active Operational Dataset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aDataset	A pointer to where the Active Operational Dataset will be placed.

Definition at line `353` of file `include/openthread/dataset.h`

otDatasetGetActiveTlvs

```
otError otDatasetGetActiveTlvs (otInstance *aInstance, otOperationalDatasetTlvs *aDataset)
```

Gets the Active Operational Dataset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aDataset	A pointer to where the Active Operational Dataset will be placed.

Definition at line `365` of file `include/openthread/dataset.h`

otDatasetSetActive

```
otError otDatasetSetActive (otInstance *aInstance, const otOperationalDataset *aDataset)
```

Sets the Active Operational Dataset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDataset	A pointer to the Active Operational Dataset.

If the dataset does not include an Active Timestamp, the dataset is only partially complete.

If Thread is enabled on a device that has a partially complete Active Dataset, the device will attempt to attach to an existing Thread network using any existing information in the dataset. Only the Thread Network Key is needed to attach to a network.

If channel is not included in the dataset, the device will send MLE Announce messages across different channels to find neighbors on other channels.

If the device successfully attaches to a Thread network, the device will then retrieve the full Active Dataset from its Parent. Note that a router-capable device will not transition to the Router or Leader roles until it has a complete Active Dataset.

Definition at line 391 of file include/openthread/dataset.h

otDatasetSetActiveTlvs

> otError otDatasetSetActiveTlvs (otInstance *aInstance, const otOperationalDatasetTlvs *aDataset)

Sets the Active Operational Dataset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDataset	A pointer to the Active Operational Dataset.

If the dataset does not include an Active Timestamp, the dataset is only partially complete.

If Thread is enabled on a device that has a partially complete Active Dataset, the device will attempt to attach to an existing Thread network using any existing information in the dataset. Only the Thread Network Key is needed to attach to a network.

If channel is not included in the dataset, the device will send MLE Announce messages across different channels to find neighbors on other channels.

If the device successfully attaches to a Thread network, the device will then retrieve the full Active Dataset from its Parent. Note that a router-capable device will not transition to the Router or Leader roles until it has a complete Active Dataset.

Definition at line 417 of file include/openthread/dataset.h

otDatasetGetPending

> otError otDatasetGetPending (otInstance *aInstance, otOperationalDataset *aDataset)

Gets the Pending Operational Dataset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aDataset	A pointer to where the Pending Operational Dataset will be placed.

Definition at line 429 of file include/openthread/dataset.h

otDatasetGetPendingTlvs

> otError otDatasetGetPendingTlvs (otInstance *aInstance, otOperationalDatasetTlvs *aDataset)

Gets the Pending Operational Dataset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aDataset	A pointer to where the Pending Operational Dataset will be placed.

Definition at line 441 of file include/openthread/dataset.h

otDatasetSetPending

> otError otDatasetSetPending (otInstance *aInstance, const otOperationalDataset *aDataset)

Sets the Pending Operational Dataset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDataset	A pointer to the Pending Operational Dataset.

Definition at line 454 of file include/openthread/dataset.h

otDatasetSetPendingTlvs

> otError otDatasetSetPendingTlvs (otInstance *aInstance, const otOperationalDatasetTlvs *aDataset)

Sets the Pending Operational Dataset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDataset	A pointer to the Pending Operational Dataset.

Definition at line 467 of file include/openthread/dataset.h

otDatasetSendMgmtActiveGet

> otError otDatasetSendMgmtActiveGet (otInstance *aInstance, const otOperationalDatasetComponents *aDatasetComponents, const uint8_t *aTlvTypes, uint8_t aLength, const otIp6Address *aAddress)

Sends MGMT_ACTIVE_GET.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDatasetComponents	A pointer to a Dataset Components structure specifying which components to request.
[in]	aTlvTypes	A pointer to array containing additional raw TLV types to be requested.
[in]	aLength	The length of `aTlvTypes`.
[in]	aAddress	A pointer to the IPv6 destination, if it is NULL, will use Leader ALOC as default.

Definition at line 482 of file include/openthread/dataset.h

otDatasetSendMgmtActiveSet

> otError otDatasetSendMgmtActiveSet (otInstance *aInstance, const otOperationalDataset *aDataset, const uint8_t *aTlvs, uint8_t aLength, otDatasetMgmtSetCallback aCallback, void *aContext)

Sends MGMT_ACTIVE_SET.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDataset	A pointer to operational dataset.
[in]	aTlvs	A pointer to TLVs.
[in]	aLength	The length of TLVs.
[in]	aCallback	A pointer to a function that is called on response reception or timeout.
[in]	aContext	A pointer to application-specific context for `aCallback`.

Definition at line `503` of file `include/openthread/dataset.h`

otDatasetSendMgmtPendingGet

```
otError otDatasetSendMgmtPendingGet (otInstance *aInstance, const otOperationalDatasetComponents
*aDatasetComponents, const uint8_t *aTlvTypes, uint8_t aLength, const otIp6Address *aAddress)
```

Sends MGMT_PENDING_GET.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDatasetComponents	A pointer to a Dataset Components structure specifying which components to request.
[in]	aTlvTypes	A pointer to array containing additional raw TLV types to be requested.
[in]	aLength	The length of `aTlvTypes`.
[in]	aAddress	A pointer to the IPv6 destination, if it is NULL, will use Leader ALOC as default.

Definition at line `523` of file `include/openthread/dataset.h`

otDatasetSendMgmtPendingSet

```
otError otDatasetSendMgmtPendingSet (otInstance *aInstance, const otOperationalDataset *aDataset, const uint8_t
*aTlvs, uint8_t aLength, otDatasetMgmtSetCallback aCallback, void *aContext)
```

Sends MGMT_PENDING_SET.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDataset	A pointer to operational dataset.
[in]	aTlvs	A pointer to TLVs.
[in]	aLength	The length of TLVs.
[in]	aCallback	A pointer to a function that is called on response reception or timeout.
[in]	aContext	A pointer to application-specific context for `aCallback`.

Definition at line `544` of file `include/openthread/dataset.h`

otDatasetGeneratePskc

> otError otDatasetGeneratePskc (const char *aPassPhrase, const otNetworkName *aNetworkName, const otExtendedPanId *aExtPanId, otPskc *aPskc)

Generates PSKc from a given pass-phrase, network name, and extended PAN ID.

Parameters

[in]	aPassPhrase	The commissioning pass-phrase.
[in]	aNetworkName	The network name for PSKc computation.
[in]	aExtPanId	The extended PAN ID for PSKc computation.
[out]	aPskc	A pointer to variable to output the generated PSKc.

PSKc is used to establish the Commissioner Session.

Definition at line `565` of file `include/openthread/dataset.h`

otNetworkNameFromString

> otError otNetworkNameFromString (otNetworkName *aNetworkName, const char *aNameString)

Sets an `otNetworkName` instance from a given null terminated C string.

Parameters

[out]	aNetworkName	A pointer to the `otNetworkName` to set.
[in]	aNameString	A name C string.

`aNameString` must follow UTF-8 encoding and the Network Name length must not be longer than `OT_NETWORK_NAME_MAX_SIZE`.

Definition at line `583` of file `include/openthread/dataset.h`

otDatasetParseTlvs

> otError otDatasetParseTlvs (const otOperationalDatasetTlvs *aDatasetTlvs, otOperationalDataset *aDataset)

Parses an Operational Dataset from a given `otOperationalDatasetTlvs`.

Parameters

[in]	aDatasetTlvs	A pointer to dataset TLVs.
[out]	aDataset	A pointer to where the dataset will be placed.

Definition at line `595` of file `include/openthread/dataset.h`

otDatasetConvertToTlvs

> otError otDatasetConvertToTlvs (const otOperationalDataset *aDataset, otOperationalDatasetTlvs *aDatasetTlvs)

Converts a given Operational Dataset to `otOperationalDatasetTlvs`.

Parameters

[in]	aDataset	An Operational dataset to convert to TLVs.
[out]	aDatasetTlvs	A pointer to dataset TLVs to return the result.

Definition at line `607` of file `include/openthread/dataset.h`

otDatasetUpdateTlvs

> otError otDatasetUpdateTlvs (const otOperationalDataset *aDataset, otOperationalDatasetTlvs *aDatasetTlvs)

Updates a given Operational Dataset.

Parameters

[in]	aDataset	Specifies the set of types and values to update.
[inout]	aDatasetTlvs	A pointer to dataset TLVs to update.

`aDataset` contains the fields to be updated and their new value.

Definition at line `622` of file `include/openthread/dataset.h`

otDatasetCreateNewNetwork

> otError otDatasetCreateNewNetwork (otInstance *aInstance, otOperationalDataset *aDataset)

For FTD only, creates a new Operational Dataset to use when forming a new network.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aDataset	The Operational Dataset.

Definition at line `62` of file `include/openthread/dataset_ftd.h`

otDatasetGetDelayTimerMinimal

> uint32_t otDatasetGetDelayTimerMinimal (otInstance *aInstance)

For FTD only, gets a minimal delay timer.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `72` of file `include/openthread/dataset_ftd.h`

otDatasetSetDelayTimerMinimal

> otError otDatasetSetDelayTimerMinimal (otInstance *aInstance, uint32_t aDelayTimerMinimal)

For FTD only, sets a minimal delay timer.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[in]	aDelayTimerMinimal	The value of minimal delay timer (in ms).

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

Definition at line 87 of file include/openthread/dataset_ftd.h

otDatasetUpdaterRequestUpdate

```
otError otDatasetUpdaterRequestUpdate (otInstance *aInstance, const otOperationalDataset *aDataset,
otDatasetUpdaterCallback aCallback, void *aContext)
```

Requests an update to Operational Dataset.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDataset	A pointer to the Dataset containing the fields to change.
[in]	aCallback	A callback to indicate when Dataset update request finishes.
[in]	aContext	An arbitrary context passed to callback.

Available when OPENTHREAD_CONFIG_DATASET_UPDATER_ENABLE is enabled.

aDataset should contain the fields to be updated and their new value. It must not contain Active or Pending Timestamp fields. The Delay field is optional, if not provided a default value (1000 ms) would be used.

Definition at line 91 of file include/openthread/dataset_updater.h

otDatasetUpdaterCancelUpdate

```
void otDatasetUpdaterCancelUpdate (otInstance *aInstance)
```

Cancels an ongoing (if any) Operational Dataset update request.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Available when OPENTHREAD_CONFIG_DATASET_UPDATER_ENABLE is enabled.

Definition at line 104 of file include/openthread/dataset_updater.h

otDatasetUpdaterIsUpdateOngoing

```
bool otDatasetUpdaterIsUpdateOngoing (otInstance *aInstance)
```

Indicates whether there is an ongoing Operation Dataset update request.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Available when OPENTHREAD_CONFIG_DATASET_UPDATER_ENABLE is enabled.

Definition at line 117 of file include/openthread/dataset_updater.h

Macro Definition Documentation

OT_NETWORK_KEY_SIZE

```
#define OT_NETWORK_KEY_SIZE
```

Value:

```
16
```

Size of the Thread Network Key (bytes)

Definition at line 56 of file include/openthread/dataset.h

OT_NETWORK_NAME_MAX_SIZE

```
#define OT_NETWORK_NAME_MAX_SIZE
```

Value:

```
16
```

Maximum size of the Thread Network Name field (bytes)

Definition at line 82 of file include/openthread/dataset.h

OT_EXT_PAN_ID_SIZE

```
#define OT_EXT_PAN_ID_SIZE
```

Value:

```
8
```

Size of a Thread PAN ID (bytes)

Definition at line 95 of file include/openthread/dataset.h

OT_MESH_LOCAL_PREFIX_SIZE

```
#define OT_MESH_LOCAL_PREFIX_SIZE
```

Value:

```
OT_IP6_PREFIX_SIZE
```

Size of the Mesh Local Prefix (bytes)

Definition at line 113 of file include/openthread/dataset.h

OT_PSKC_MAX_SIZE

```
#define OT_PSKC_MAX_SIZE
```

Value:

```
16
```

Maximum size of the PSKc (bytes)

Definition at line 121 of file include/openthread/dataset.h

OT_CHANNEL_1_MASK

```
#define OT_CHANNEL_1_MASK
```

Value:

```
(1 << 1)
```

Channel 1.

Definition at line 171 of file include/openthread/dataset.h

OT_CHANNEL_2_MASK

```
#define OT_CHANNEL_2_MASK
```

Value:

```
(1 << 2)
```

Channel 2.

Definition at line 172 of file include/openthread/dataset.h

OT_CHANNEL_3_MASK

```
#define OT_CHANNEL_3_MASK
```

Value:

```
(1 << 3)
```

Channel 3.

Definition at line 173 of file include/openthread/dataset.h

OT_CHANNEL_4_MASK

```
#define OT_CHANNEL_4_MASK
```

Value:

```
(1 << 4)
```

Channel 4.

Definition at line 174 of file include/openthread/dataset.h

OT_CHANNEL_5_MASK

```
#define OT_CHANNEL_5_MASK
```

Value:

```
(1 << 5)
```

Channel 5.

Definition at line 175 of file include/openthread/dataset.h

OT_CHANNEL_6_MASK

```
#define OT_CHANNEL_6_MASK
```

Value:

```
(1 << 6)
```

Channel 6.

Definition at line 176 of file include/openthread/dataset.h

OT_CHANNEL_7_MASK

```
#define OT_CHANNEL_7_MASK
```

Value:

```
(1 << 7)
```

Channel 7.

Definition at line 177 of file include/openthread/dataset.h

OT_CHANNEL_8_MASK

```
#define OT_CHANNEL_8_MASK
```

Value:

```
(1 << 8)
```

Channel 8.

Definition at line 178 of file include/openthread/dataset.h

OT_CHANNEL_9_MASK

```
#define OT_CHANNEL_9_MASK
```

Value:

```
(1 << 9)
```

Channel 9.

Definition at line 179 of file include/openthread/dataset.h

OT_CHANNEL_10_MASK

```
#define OT_CHANNEL_10_MASK
```

Value:

```
(1 << 10)
```

Channel 10.

Definition at line 180 of file include/openthread/dataset.h

OT_CHANNEL_11_MASK

```
#define OT_CHANNEL_11_MASK
```

Value:

```
(1 << 11)
```

Channel 11.

Definition at line 181 of file include/openthread/dataset.h

OT_CHANNEL_12_MASK

```
#define OT_CHANNEL_12_MASK
```

Value:

```
(1 << 12)
```

Channel 12.

Definition at line 182 of file include/openthread/dataset.h

OT_CHANNEL_13_MASK

```
#define OT_CHANNEL_13_MASK
```

Value:

```
(1 << 13)
```

Channel 13.

Definition at line 183 of file include/openthread/dataset.h

OT_CHANNEL_14_MASK

```
#define OT_CHANNEL_14_MASK
```

Value:

```
(1 << 14)
```

Channel 14.

Definition at line 184 of file include/openthread/dataset.h

OT_CHANNEL_15_MASK

```
#define OT_CHANNEL_15_MASK
```

Value:

```
(1 << 15)
```

Channel 15.

Definition at line 185 of file include/openthread/dataset.h

OT_CHANNEL_16_MASK

```
#define OT_CHANNEL_16_MASK
```

Value:

```
(1 << 16)
```

Channel 16.

Definition at line 186 of file include/openthread/dataset.h

OT_CHANNEL_17_MASK

```
#define OT_CHANNEL_17_MASK
```

Value:

```
(1 << 17)
```

Channel 17.

Definition at line 187 of file include/openthread/dataset.h

OT_CHANNEL_18_MASK

```
#define OT_CHANNEL_18_MASK
```

Value:

```
(1 << 18)
```

Channel 18.

Definition at line 188 of file include/openthread/dataset.h

OT_CHANNEL_19_MASK

```
#define OT_CHANNEL_19_MASK
```

Value:

```
(1 << 19)
```

Channel 19.

Definition at line 189 of file include/openthread/dataset.h

OT_CHANNEL_20_MASK

```
#define OT_CHANNEL_20_MASK
```

Value:

```
(1 << 20)
```

Channel 20.

Definition at line 190 of file include/openthread/dataset.h

OT_CHANNEL_21_MASK

```
#define OT_CHANNEL_21_MASK
```

Value:

```
(1 << 21)
```

Channel 21.

Definition at line 191 of file include/openthread/dataset.h

OT_CHANNEL_22_MASK

```
#define OT_CHANNEL_22_MASK
```

Value:

```
(1 << 22)
```

Channel 22.

Definition at line 192 of file include/openthread/dataset.h

OT_CHANNEL_23_MASK

```
#define OT_CHANNEL_23_MASK
```

Value:

```
(1 << 23)
```

Channel 23.

Definition at line 193 of file include/openthread/dataset.h

OT_CHANNEL_24_MASK

```
#define OT_CHANNEL_24_MASK
```

Value:

```
(1 << 24)
```

Channel 24.

Definition at line 194 of file include/openthread/dataset.h

OT_CHANNEL_25_MASK

```
#define OT_CHANNEL_25_MASK
```

Value:

```
(1 << 25)
```

Channel 25.

Definition at line 195 of file include/openthread/dataset.h

OT_CHANNEL_26_MASK

```
#define OT_CHANNEL_26_MASK
```

Value:

```
(1 << 26)
```

Channel 26.

Definition at line 196 of file include/openthread/dataset.h

OT_OPERATIONAL_DATASET_MAX_LENGTH

```
#define OT_OPERATIONAL_DATASET_MAX_LENGTH
```

Value:

```
254
```

Maximum length of Operational Dataset in bytes.

Definition at line 256 of file include/openthread/dataset.h

otNetworkKey

Represents a Thread Network Key.

Public Attributes

uint8_t	m8 Byte values.

Public Attribute Documentation

m8

```
uint8_t otNetworkKey::m8[OT_NETWORK_KEY_SIZE]
```

Byte values.

Definition at line 67 of file include/openthread/dataset.h

otNetworkName

Represents a Network Name.

The `otNetworkName` is a null terminated C string (i.e., `m8` char array MUST end with null char `\0`).

Public Attributes

char	m8 Byte values. The `+1` is for null char.

Public Attribute Documentation

m8

```
char otNetworkName::m8[OT_NETWORK_NAME_MAX_SIZE+1]
```

Byte values. The `+1` is for null char.

Definition at line `92` of file `include/openthread/dataset.h`

otExtendedPanId

Represents an Extended PAN ID.

Public Attributes

uint8_t	m8 Byte values.

Public Attribute Documentation

m8

```
uint8_t otExtendedPanId::m8[OT_EXT_PAN_ID_SIZE]
```

Byte values.

Definition at line 104 of file include/openthread/dataset.h

otPskc

Represents PSKc.

Public Attributes

uint8_t	m8 Byte values.

Public Attribute Documentation

m8

```
uint8_t otPskc::m8[OT_PSKC_MAX_SIZE]
```

Byte values.

Definition at line 130 of file include/openthread/dataset.h

otSecurityPolicy

Represent Security Policy.

Public Attributes

uint16_t	mRotationTime The value for thrKeyRotation in units of hours.
bool	mObtainNetworkKeyEnabled Obtaining the Network Key for out-of-band commissioning is enabled.
bool	mNativeCommissioningEnabled Native Commissioning using PSKc is allowed.
bool	mRoutersEnabled Thread 1.0/1.1.x Routers are enabled.
bool	mExternalCommissioningEnabled External Commissioner authentication is allowed.
bool	mCommercialCommissioningEnabled Commercial Commissioning is enabled.
bool	mAutonomousEnrollmentEnabled Autonomous Enrollment is enabled.
bool	mNetworkKeyProvisioningEnabled Network Key Provisioning is enabled.
bool	mTobleLinkEnabled ToBLE link is enabled.
bool	mNonCcmRoutersEnabled Non-CCM Routers enabled.
uint8_t	mVersionThresholdForRouting Version-threshold for Routing.

Public Attribute Documentation

mRotationTime

```
uint16_t otSecurityPolicy::mRotationTime
```

The value for thrKeyRotation in units of hours.

Definition at line 151 of file include/openthread/dataset.h

mObtainNetworkKeyEnabled

```
bool otSecurityPolicy::mObtainNetworkKeyEnabled
```

Obtaining the Network Key for out-of-band commissioning is enabled.

Definition at line 153 of file include/openthread/dataset.h

mNativeCommissioningEnabled

```
bool otSecurityPolicy::mNativeCommissioningEnabled
```

Native Commissioning using PSKc is allowed.

Definition at line 154 of file include/openthread/dataset.h

mRoutersEnabled

```
bool otSecurityPolicy::mRoutersEnabled
```

Thread 1.0/1.1.x Routers are enabled.

Definition at line 155 of file include/openthread/dataset.h

mExternalCommissioningEnabled

```
bool otSecurityPolicy::mExternalCommissioningEnabled
```

External Commissioner authentication is allowed.

Definition at line 156 of file include/openthread/dataset.h

mCommercialCommissioningEnabled

```
bool otSecurityPolicy::mCommercialCommissioningEnabled
```

Commercial Commissioning is enabled.

Definition at line 157 of file include/openthread/dataset.h

mAutonomousEnrollmentEnabled

```
bool otSecurityPolicy::mAutonomousEnrollmentEnabled
```

Autonomous Enrollment is enabled.

Definition at line 158 of file include/openthread/dataset.h

mNetworkKeyProvisioningEnabled

```
bool otSecurityPolicy::mNetworkKeyProvisioningEnabled
```

Network Key Provisioning is enabled.

Definition at line 159 of file include/openthread/dataset.h

mTobleLinkEnabled

```
bool otSecurityPolicy::mTobleLinkEnabled
```

ToBLE link is enabled.

Definition at line `160` of file `include/openthread/dataset.h`

mNonCcmRoutersEnabled

```
bool otSecurityPolicy::mNonCcmRoutersEnabled
```

Non-CCM Routers enabled.

Definition at line `161` of file `include/openthread/dataset.h`

mVersionThresholdForRouting

```
uint8_t otSecurityPolicy::mVersionThresholdForRouting
```

Version-threshold for Routing.

Definition at line `162` of file `include/openthread/dataset.h`

otOperationalDatasetComponents

Represents presence of different components in Active or Pending Operational Dataset.

Public Attributes

bool mIsActiveTimestampPresent
TRUE if Active Timestamp is present, FALSE otherwise.

bool mIsPendingTimestampPresent
TRUE if Pending Timestamp is present, FALSE otherwise.

bool mIsNetworkKeyPresent
TRUE if Network Key is present, FALSE otherwise.

bool mIsNetworkNamePresent
TRUE if Network Name is present, FALSE otherwise.

bool mIsExtendedPanIdPresent
TRUE if Extended PAN ID is present, FALSE otherwise.

bool mIsMeshLocalPrefixPresent
TRUE if Mesh Local Prefix is present, FALSE otherwise.

bool mIsDelayPresent
TRUE if Delay Timer is present, FALSE otherwise.

bool mIsPanIdPresent
TRUE if PAN ID is present, FALSE otherwise.

bool mIsChannelPresent
TRUE if Channel is present, FALSE otherwise.

bool mIsPskcPresent
TRUE if PSKc is present, FALSE otherwise.

bool mIsSecurityPolicyPresent
TRUE if Security Policy is present, FALSE otherwise.

bool mIsChannelMaskPresent
TRUE if Channel Mask is present, FALSE otherwise.

Public Attribute Documentation

mIsActiveTimestampPresent

> bool otOperationalDatasetComponents::mIsActiveTimestampPresent

TRUE if Active Timestamp is present, FALSE otherwise.

Definition at line 204 of file include/openthread/dataset.h

mIsPendingTimestampPresent

bool otOperationalDatasetComponents::mIsPendingTimestampPresent

TRUE if Pending Timestamp is present, FALSE otherwise.

Definition at line 205 of file include/openthread/dataset.h

mIsNetworkKeyPresent

bool otOperationalDatasetComponents::mIsNetworkKeyPresent

TRUE if Network Key is present, FALSE otherwise.

Definition at line 206 of file include/openthread/dataset.h

mIsNetworkNamePresent

bool otOperationalDatasetComponents::mIsNetworkNamePresent

TRUE if Network Name is present, FALSE otherwise.

Definition at line 207 of file include/openthread/dataset.h

mIsExtendedPanIdPresent

bool otOperationalDatasetComponents::mIsExtendedPanIdPresent

TRUE if Extended PAN ID is present, FALSE otherwise.

Definition at line 208 of file include/openthread/dataset.h

mIsMeshLocalPrefixPresent

bool otOperationalDatasetComponents::mIsMeshLocalPrefixPresent

TRUE if Mesh Local Prefix is present, FALSE otherwise.

Definition at line 209 of file include/openthread/dataset.h

mIsDelayPresent

bool otOperationalDatasetComponents::mIsDelayPresent

TRUE if Delay Timer is present, FALSE otherwise.

Definition at line 210 of file include/openthread/dataset.h

mIsPanIdPresent

bool otOperationalDatasetComponents::mIsPanIdPresent

TRUE if PAN ID is present, FALSE otherwise.

Definition at line 211 of file include/openthread/dataset.h

mIsChannelPresent

bool otOperationalDatasetComponents::mIsChannelPresent

TRUE if Channel is present, FALSE otherwise.

Definition at line 212 of file include/openthread/dataset.h

mIsPskcPresent

bool otOperationalDatasetComponents::mIsPskcPresent

TRUE if PSKc is present, FALSE otherwise.

Definition at line 213 of file include/openthread/dataset.h

mIsSecurityPolicyPresent

bool otOperationalDatasetComponents::mIsSecurityPolicyPresent

TRUE if Security Policy is present, FALSE otherwise.

Definition at line 214 of file include/openthread/dataset.h

mIsChannelMaskPresent

bool otOperationalDatasetComponents::mIsChannelMaskPresent

TRUE if Channel Mask is present, FALSE otherwise.

Definition at line 215 of file include/openthread/dataset.h

otTimestamp

Represents a Thread Dataset timestamp component.

Public Attributes

uint64_t	mSeconds
uint16_t	mTicks
bool	mAuthoritative

Public Attribute Documentation

mSeconds

```
uint64_t otTimestamp::mSeconds
```

Definition at line 224 of file include/openthread/dataset.h

mTicks

```
uint16_t otTimestamp::mTicks
```

Definition at line 225 of file include/openthread/dataset.h

mAuthoritative

```
bool otTimestamp::mAuthoritative
```

Definition at line 226 of file include/openthread/dataset.h

otOperationalDataset

Represents an Active or Pending Operational Dataset.

Components in Dataset are optional. `mComponents` structure specifies which components are present in the Dataset.

Public Attributes

otTimestamp	mActiveTimestamp Active Timestamp.
otTimestamp	mPendingTimestamp Pending Timestamp.
otNetworkKey	mNetworkKey Network Key.
otNetworkName	mNetworkName Network Name.
otExtendedPanId	mExtendedPanId Extended PAN ID.
otMeshLocalPrefix	mMeshLocalPrefix Mesh Local Prefix.
uint32_t	mDelay Delay Timer.
otPanId	mPanId PAN ID.
uint16_t	mChannel Channel.
otPskc	mPskc PSKc.
otSecurityPolicy	mSecurityPolicy Security Policy.
otChannelMask	mChannelMask Channel Mask.
otOperationalDatasetComponents	mComponents Specifies which components are set in the Dataset.

Public Attribute Documentation

mActiveTimestamp

> otTimestamp otOperationalDataset::mActiveTimestamp

Active Timestamp.

Definition at line `237` of file `include/openthread/dataset.h`

mPendingTimestamp

> otTimestamp otOperationalDataset::mPendingTimestamp

Pending Timestamp.

Definition at line 238 of file include/openthread/dataset.h

mNetworkKey

> otNetworkKey otOperationalDataset::mNetworkKey

Network Key.

Definition at line 239 of file include/openthread/dataset.h

mNetworkName

> otNetworkName otOperationalDataset::mNetworkName

Network Name.

Definition at line 240 of file include/openthread/dataset.h

mExtendedPanId

> otExtendedPanId otOperationalDataset::mExtendedPanId

Extended PAN ID.

Definition at line 241 of file include/openthread/dataset.h

mMeshLocalPrefix

> otMeshLocalPrefix otOperationalDataset::mMeshLocalPrefix

Mesh Local Prefix.

Definition at line 242 of file include/openthread/dataset.h

mDelay

> uint32_t otOperationalDataset::mDelay

Delay Timer.

Definition at line 243 of file include/openthread/dataset.h

mPanId

> otPanId otOperationalDataset::mPanId

PAN ID.

Definition at line 244 of file include/openthread/dataset.h

mChannel

> uint16_t otOperationalDataset::mChannel

Channel.

Definition at line 245 of file include/openthread/dataset.h

mPskc

> otPskc otOperationalDataset::mPskc

PSKc.

Definition at line 246 of file include/openthread/dataset.h

mSecurityPolicy

> otSecurityPolicy otOperationalDataset::mSecurityPolicy

Security Policy.

Definition at line 247 of file include/openthread/dataset.h

mChannelMask

> otChannelMask otOperationalDataset::mChannelMask

Channel Mask.

Definition at line 248 of file include/openthread/dataset.h

mComponents

> otOperationalDatasetComponents otOperationalDataset::mComponents

Specifies which components are set in the Dataset.

Definition at line 249 of file include/openthread/dataset.h

otOperationalDatasetTlvs

Represents an Active or Pending Operational Dataset.

The Operational Dataset is TLV encoded as specified by Thread.

Public Attributes

uint8_t	mTlvs Operational Dataset TLVs.
uint8_t	mLength Size of Operational Dataset in bytes.

Public Attribute Documentation

mTlvs

```
uint8_t otOperationalDatasetTlvs::mTlvs[OT_OPERATIONAL_DATASET_MAX_LENGTH]
```

Operational Dataset TLVs.

Definition at line 266 of file include/openthread/dataset.h

mLength

```
uint8_t otOperationalDatasetTlvs::mLength
```

Size of Operational Dataset in bytes.

Definition at line 267 of file include/openthread/dataset.h

Router/Leader

Router/Leader

This module includes functions for Thread Routers and Leaders.

Modules

otChildInfo

otCacheEntryInfo

otCacheEntryIterator

otDeviceProperties

otNeighborTableEntryInfo

Enumerations

enum otCacheEntryState {

OT_CACHE_ENTRY_STATE_CACHED = 0
OT_CACHE_ENTRY_STATE_SNOOPED = 1
OT_CACHE_ENTRY_STATE_QUERY = 2
OT_CACHE_ENTRY_STATE_RETRY_QUERY = 3

}

Defines the EID cache entry state.

enum otPowerSupply {

OT_POWER_SUPPLY_BATTERY = 0
OT_POWER_SUPPLY_EXTERNAL = 1
OT_POWER_SUPPLY_EXTERNAL_STABLE = 2
OT_POWER_SUPPLY_EXTERNAL_UNSTABLE = 3

}

Represents the power supply property on a device.

enum otNeighborTableEvent {

OT_NEIGHBOR_TABLE_EVENT_CHILD_ADDED
OT_NEIGHBOR_TABLE_EVENT_CHILD_REMOVED
OT_NEIGHBOR_TABLE_EVENT_CHILD_MODE_CHANGED
OT_NEIGHBOR_TABLE_EVENT_ROUTER_ADDED
OT_NEIGHBOR_TABLE_EVENT_ROUTER_REMOVED

}

Defines the constants used in `otNeighborTableCallback` to indicate changes in neighbor table.

Typedefs

typedef uint16_t otChildIp6AddressIterator

Used to iterate through IPv6 addresses of a Thread Child entry.

typedef enum otCacheEntryState otCacheEntryState

Defines the EID cache entry state.

typedef struct otCacheEntryInfo — otCacheEntryInfo
Represents an EID cache entry.

typedef struct otCacheEntryIterator — otCacheEntryIterator
Represents an iterator used for iterating through the EID cache table entries.

typedef struct otDeviceProperties — otDeviceProperties
Represents the device properties which are used for calculating the local leader weight on a device.

typedef void(* otNeighborTableCallback)(otNeighborTableEvent aEvent, const otNeighborTableEntryInfo *aEntryInfo)
Pointer is called to notify that there is a change in the neighbor table.

Functions

uint16_t otThreadGetMaxAllowedChildren(otInstance *aInstance)
Gets the maximum number of children currently allowed.

otError otThreadSetMaxAllowedChildren(otInstance *aInstance, uint16_t aMaxChildren)
Sets the maximum number of children currently allowed.

bool otThreadIsRouterEligible(otInstance *aInstance)
Indicates whether or not the device is router-eligible.

otError otThreadSetRouterEligible(otInstance *aInstance, bool aEligible)
Sets whether or not the device is router-eligible.

otError otThreadSetPreferredRouterId(otInstance *aInstance, uint8_t aRouterId)
Set the preferred Router Id.

const otDeviceProperties * otThreadGetDeviceProperties(otInstance *aInstance)
Get the current device properties.

void otThreadSetDeviceProperties(otInstance *aInstance, const otDeviceProperties *aDeviceProperties)
Set the device properties which are then used to determine and set the Leader Weight.

uint8_t otThreadGetLocalLeaderWeight(otInstance *aInstance)
Gets the Thread Leader Weight used when operating in the Leader role.

void otThreadSetLocalLeaderWeight(otInstance *aInstance, uint8_t aWeight)
Sets the Thread Leader Weight used when operating in the Leader role.

uint32_t otThreadGetPreferredLeaderPartitionId(otInstance *aInstance)
Get the preferred Thread Leader Partition Id used when operating in the Leader role.

void otThreadSetPreferredLeaderPartitionId(otInstance *aInstance, uint32_t aPartitionId)
Set the preferred Thread Leader Partition Id used when operating in the Leader role.

uint16_t otThreadGetJoinerUdpPort(otInstance *aInstance)
Gets the Joiner UDP Port.

otError otThreadSetJoinerUdpPort(otInstance *aInstance, uint16_t aJoinerUdpPort)
Sets the Joiner UDP Port.

void otThreadSetSteeringData(otInstance *aInstance, const otExtAddress *aExtAddress)
Set Steering data out of band.

uint32_t otThreadGetContextIdReuseDelay(otInstance *aInstance)
Get the CONTEXT_ID_REUSE_DELAY parameter used in the Leader role.

void otThreadSetContextIdReuseDelay(otInstance *aInstance, uint32_t aDelay)
Set the CONTEXT_ID_REUSE_DELAY parameter used in the Leader role.

uint8_t otThreadGetNetworkIdTimeout(otInstance *aInstance)
Get the `NETWORK_ID_TIMEOUT` parameter.

void otThreadSetNetworkIdTimeout(otInstance *aInstance, uint8_t aTimeout)
Set the `NETWORK_ID_TIMEOUT` parameter.

uint8_t otThreadGetRouterUpgradeThreshold(otInstance *aInstance)
Get the ROUTER_UPGRADE_THRESHOLD parameter used in the REED role.

void otThreadSetRouterUpgradeThreshold(otInstance *aInstance, uint8_t aThreshold)
Set the ROUTER_UPGRADE_THRESHOLD parameter used in the Leader role.

uint8_t otThreadGetChildRouterLinks(otInstance *aInstance)
Get the MLE_CHILD_ROUTER_LINKS parameter used in the REED role.

otError otThreadSetChildRouterLinks(otInstance *aInstance, uint8_t aChildRouterLinks)
Set the MLE_CHILD_ROUTER_LINKS parameter used in the REED role.

otError otThreadReleaseRouterId(otInstance *aInstance, uint8_t aRouterId)
Release a Router ID that has been allocated by the device in the Leader role.

otError otThreadBecomeRouter(otInstance *aInstance)
Attempt to become a router.

otError otThreadBecomeLeader(otInstance *aInstance)
Become a leader and start a new partition.

uint8_t otThreadGetRouterDowngradeThreshold(otInstance *aInstance)
Get the ROUTER_DOWNGRADE_THRESHOLD parameter used in the Router role.

void otThreadSetRouterDowngradeThreshold(otInstance *aInstance, uint8_t aThreshold)
Set the ROUTER_DOWNGRADE_THRESHOLD parameter used in the Leader role.

uint8_t otThreadGetRouterSelectionJitter(otInstance *aInstance)
Get the ROUTER_SELECTION_JITTER parameter used in the REED/Router role.

void otThreadSetRouterSelectionJitter(otInstance *aInstance, uint8_t aRouterJitter)
Set the ROUTER_SELECTION_JITTER parameter used in the REED/Router role.

otError otThreadGetChildInfoById(otInstance *aInstance, uint16_t aChildId, otChildInfo *aChildInfo)
Gets diagnostic information for an attached Child by its Child ID or RLOC16.

otError otThreadGetChildInfoByIndex(otInstance *aInstance, uint16_t aChildIndex, otChildInfo *aChildInfo)
The function retains diagnostic information for an attached Child by the internal table index.

otError otThreadGetChildNextIp6Address(otInstance *aInstance, uint16_t aChildIndex, otChildIp6AddressIterator *aIterator, otIp6Address *aAddress)
Gets the next IPv6 address (using an iterator) for a given child.

uint8_t otThreadGetRouterIdSequence(otInstance *aInstance)
Get the current Router ID Sequence.

uint8_t otThreadGetMaxRouterId(otInstance *aInstance)
The function returns the maximum allowed router ID.

otError otThreadGetRouterInfo(otInstance *aInstance, uint16_t aRouterId, otRouterInfo *aRouterInfo)
The function retains diagnostic information for a given Thread Router.

otError	otThreadGetNextCacheEntry(otInstance *aInstance, otCacheEntryInfo *aEntryInfo, otCacheEntryIterator *aIterator) Gets the next EID cache entry (using an iterator).
void	otThreadGetPskc(otInstance *aInstance, otPskc *aPskc) Get the Thread PSKc.
otPskcRef	otThreadGetPskcRef(otInstance *aInstance) Get Key Reference to Thread PSKc stored.
otError	otThreadSetPskc(otInstance *aInstance, const otPskc *aPskc) Set the Thread PSKc.
otError	otThreadSetPskcRef(otInstance *aInstance, otPskcRef aKeyRef) Set the Key Reference to the Thread PSKc.
int8_t	otThreadGetParentPriority(otInstance *aInstance) Get the assigned parent priority.
otError	otThreadSetParentPriority(otInstance *aInstance, int8_t aParentPriority) Set the parent priority.
uint8_t	otThreadGetMaxChildIpAddresses(otInstance *aInstance) Gets the maximum number of IP addresses that each MTD child may register with this device as parent.
otError	otThreadSetMaxChildIpAddresses(otInstance *aInstance, uint8_t aMaxIpAddresses) Sets or restores the maximum number of IP addresses that each MTD child may register with this device as parent.
void	otThreadRegisterNeighborTableCallback(otInstance *aInstance, otNeighborTableCallback aCallback) Registers a neighbor table callback function.
void	otThreadSetCcmEnabled(otInstance *aInstance, bool aEnabled) Sets whether the device was commissioned using CCM.
void	otThreadSetThreadVersionCheckEnabled(otInstance *aInstance, bool aEnabled) Sets whether the Security Policy TLV version-threshold for routing (VR field) is enabled.
void	otThreadGetRouterIdRange(otInstance *aInstance, uint8_t *aMinRouterId, uint8_t *aMaxRouterId) Gets the range of router IDs that are allowed to assign to nodes within the thread network.
otError	otThreadSetRouterIdRange(otInstance *aInstance, uint8_t aMinRouterId, uint8_t aMaxRouterId) Sets the range of router IDs that are allowed to assign to nodes within the thread network.
uint32_t	otThreadGetAdvertisementTrickleIntervalMax(otInstance *aInstance) Gets the current Interval Max value used by Advertisement trickle timer.
bool	otThreadIsRouterIdAllocated(otInstance *aInstance, uint8_t aRouterId) Indicates whether or not a Router ID is currently allocated.
void	otThreadGetNextHopAndPathCost(otInstance *aInstance, uint16_t aDestRloc16, uint16_t *aNextHopRloc16, uint8_t *aPathCost) Gets the next hop and path cost towards a given RLOC16 destination.

Macros

#define	OT_CHILD_IP6_ADDRESS_ITERATOR_INIT 0 Initializer for otChildIp6AddressIterator.

Enumeration Documentation

otCacheEntryState

> otCacheEntryState

Defines the EID cache entry state.

Enumerator	
OT_CACHE_ENTRY_STATE_CACHED	
OT_CACHE_ENTRY_STATE_SNOOPED	
OT_CACHE_ENTRY_STATE_QUERY	
OT_CACHE_ENTRY_STATE_RETRY_QUERY	

Definition at line `89` of file `include/openthread/thread_ftd.h`

otPowerSupply

> otPowerSupply

Represents the power supply property on a device.

This is used as a property in `otDeviceProperties` to calculate the leader weight.

Enumerator	
OT_POWER_SUPPLY_BATTERY	Battery powered.
OT_POWER_SUPPLY_EXTERNAL	Externally powered (mains-powered).
OT_POWER_SUPPLY_EXTERNAL_STABLE	Stable external power with a battery backup or UPS.
OT_POWER_SUPPLY_EXTERNAL_UNSTABLE	Potentially unstable ext power (e.g. light bulb powered via a switch).

Definition at line `207` of file `include/openthread/thread_ftd.h`

otNeighborTableEvent

> otNeighborTableEvent

Defines the constants used in `otNeighborTableCallback` to indicate changes in neighbor table.

Enumerator	
OT_NEIGHBOR_TABLE_EVENT_CHILD_ADDED	A child is being added.
OT_NEIGHBOR_TABLE_EVENT_CHILD_REMOVED	A child is being removed.
OT_NEIGHBOR_TABLE_EVENT_CHILD_MODE_CHANGED	An existing child's mode is changed.
OT_NEIGHBOR_TABLE_EVENT_ROUTER_ADDED	A router is being added.
OT_NEIGHBOR_TABLE_EVENT_ROUTER_REMOVED	A router is being removed.

Definition at line `782` of file `include/openthread/thread_ftd.h`

Typedef Documentation

otChildIp6AddressIterator

> typedef uint16_t otChildIp6AddressIterator

Used to iterate through IPv6 addresses of a Thread Child entry.

Definition at line `83` of file `include/openthread/thread_ftd.h`

otCacheEntryState

```
typedef enum otCacheEntryState otCacheEntryState
```

Defines the EID cache entry state.

Definition at line `95` of file `include/openthread/thread_ftd.h`

otCacheEntryInfo

```
typedef struct otCacheEntryInfo otCacheEntryInfo
```

Represents an EID cache entry.

Definition at line `113` of file `include/openthread/thread_ftd.h`

otCacheEntryIterator

```
typedef struct otCacheEntryIterator otCacheEntryIterator
```

Represents an iterator used for iterating through the EID cache table entries.

To initialize the iterator and start from the first entry in the cache table, set all its fields in the structure to zero (e.g., `memset` the iterator to zero).

Definition at line `125` of file `include/openthread/thread_ftd.h`

otDeviceProperties

```
typedef struct otDeviceProperties otDeviceProperties
```

Represents the device properties which are used for calculating the local leader weight on a device.

The parameters are set based on device's capability, whether acting as border router, its power supply config, etc.

`mIsUnstable` indicates operational stability of device and is determined via a vendor specific mechanism. It can include the following cases:

- Device internally detects that it loses external power supply more often than usual. What is usual is determined by the vendor.
- Device internally detects that it reboots more often than usual. What is usual is determined by the vendor.

Definition at line `235` of file `include/openthread/thread_ftd.h`

otNeighborTableCallback

typedef void(* otNeighborTableCallback) (otNeighborTableEvent aEvent, const otNeighborTableEntryInfo *aEntryInfo))
(otNeighborTableEvent aEvent, const otNeighborTableEntryInfo *aEntryInfo)

Pointer is called to notify that there is a change in the neighbor table.

Parameters

[in]	aEvent	A event flag.
[in]	aEntryInfo	A pointer to table entry info.

Definition at line 813 of file include/openthread/thread_ftd.h

Function Documentation

otThreadGetMaxAllowedChildren

uint16_t otThreadGetMaxAllowedChildren (otInstance *aInstance)

Gets the maximum number of children currently allowed.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The maximum number of children currently allowed.

See Also

- otThreadSetMaxAllowedChildren

Definition at line 137 of file include/openthread/thread_ftd.h

otThreadSetMaxAllowedChildren

otError otThreadSetMaxAllowedChildren (otInstance *aInstance, uint16_t aMaxChildren)

Sets the maximum number of children currently allowed.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMaxChildren	The maximum allowed children.

This parameter can only be set when Thread protocol operation has been stopped.

See Also

- otThreadGetMaxAllowedChildren

Definition at line 154 of file include/openthread/thread_ftd.h

otThreadIsRouterEligible

bool otThreadIsRouterEligible (otInstance *aInstance)

Indicates whether or not the device is router-eligible.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `165` of file `include/openthread/thread_ftd.h`

otThreadSetRouterEligible

```
otError otThreadSetRouterEligible (otInstance *aInstance, bool aEligible)
```

Sets whether or not the device is router-eligible.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEligible	TRUE to configure the device as router-eligible, FALSE otherwise.

If `aEligible` is false and the device is currently operating as a router, this call will cause the device to detach and attempt to reattach as a child.

Definition at line `180` of file `include/openthread/thread_ftd.h`

otThreadSetPreferredRouterId

```
otError otThreadSetPreferredRouterId (otInstance *aInstance, uint8_t aRouterId)
```

Set the preferred Router Id.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aRouterId	The preferred Router Id.

Upon becoming a router/leader the node attempts to use this Router Id. If the preferred Router Id is not set or if it can not be used, a randomly generated router id is picked. This property can be set only when the device role is either detached or disabled.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

Definition at line `199` of file `include/openthread/thread_ftd.h`

otThreadGetDeviceProperties

```
const otDeviceProperties * otThreadGetDeviceProperties (otInstance *aInstance)
```

Get the current device properties.

Parameters

N/A	aInstance	

Requires `OPENTHREAD_CONFIG_MLE_DEVICE_PROPERTY_LEADER_WEIGHT_ENABLE`.

Returns

- The device properties `otDeviceProperties`.

Definition at line `245` of file `include/openthread/thread_ftd.h`

otThreadSetDeviceProperties

```
void otThreadSetDeviceProperties (otInstance *aInstance, const otDeviceProperties *aDeviceProperties)
```

Set the device properties which are then used to determine and set the Leader Weight.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDeviceProperties	The device properties.

Requires `OPENTHREAD_CONFIG_MLE_DEVICE_PROPERTY_LEADER_WEIGHT_ENABLE`.

Definition at line `256` of file `include/openthread/thread_ftd.h`

otThreadGetLocalLeaderWeight

```
uint8_t otThreadGetLocalLeaderWeight (otInstance *aInstance)
```

Gets the Thread Leader Weight used when operating in the Leader role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Thread Leader Weight value.

See Also

- otThreadSetLeaderWeight
- otThreadSetDeviceProperties

Definition at line `269` of file `include/openthread/thread_ftd.h`

otThreadSetLocalLeaderWeight

```
void otThreadSetLocalLeaderWeight (otInstance *aInstance, uint8_t aWeight)
```

Sets the Thread Leader Weight used when operating in the Leader role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aWeight	The Thread Leader Weight value.

Directly sets the Leader Weight to the new value, replacing its previous value (which may have been determined from the current `otDeviceProperties`).

See Also

- otThreadGetLeaderWeight

Definition at line 283 of file include/openthread/thread_ftd.h

otThreadGetPreferredLeaderPartitionId

```
uint32_t otThreadGetPreferredLeaderPartitionId (otInstance *aInstance)
```

Get the preferred Thread Leader Partition Id used when operating in the Leader role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Thread Leader Partition Id value.

Definition at line 293 of file include/openthread/thread_ftd.h

otThreadSetPreferredLeaderPartitionId

```
void otThreadSetPreferredLeaderPartitionId (otInstance *aInstance, uint32_t aPartitionId)
```

Set the preferred Thread Leader Partition Id used when operating in the Leader role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPartitionId	The Thread Leader Partition Id value.

Definition at line 302 of file include/openthread/thread_ftd.h

otThreadGetJoinerUdpPort

```
uint16_t otThreadGetJoinerUdpPort (otInstance *aInstance)
```

Gets the Joiner UDP Port.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Joiner UDP Port number.

See Also

- otThreadSetJoinerUdpPort

Definition at line 314 of file include/openthread/thread_ftd.h

otThreadSetJoinerUdpPort

```
otError otThreadSetJoinerUdpPort (otInstance *aInstance, uint16_t aJoinerUdpPort)
```

Sets the Joiner UDP Port.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aJoinerUdpPort	The Joiner UDP Port number.

See Also

- otThreadGetJoinerUdpPort

Definition at line `327` of file `include/openthread/thread_ftd.h`

otThreadSetSteeringData

```
void otThreadSetSteeringData (otInstance *aInstance, const otExtAddress *aExtAddress)
```

Set Steering data out of band.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aExtAddress	Address used to update the steering data. All zeros to clear the steering data (no steering data). All 0xFFs to set steering data/bloom filter to accept/allow all. A specific EUI64 which is then added to current steering data/bloom filter.

Configuration option `OPENTHREAD_CONFIG_MLE_STEERING_DATA_SET_OOB_ENABLE` should be set to enable setting of steering data out of band.

Definition at line `342` of file `include/openthread/thread_ftd.h`

otThreadGetContextIdReuseDelay

```
uint32_t otThreadGetContextIdReuseDelay (otInstance *aInstance)
```

Get the CONTEXT_ID_REUSE_DELAY parameter used in the Leader role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The CONTEXT_ID_REUSE_DELAY value.

See Also

- otThreadSetContextIdReuseDelay

Definition at line `354` of file `include/openthread/thread_ftd.h`

otThreadSetContextIdReuseDelay

```
void otThreadSetContextIdReuseDelay (otInstance *aInstance, uint32_t aDelay)
```

Set the CONTEXT_ID_REUSE_DELAY parameter used in the Leader role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDelay	The CONTEXT_ID_REUSE_DELAY value.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

See Also

- otThreadGetContextIdReuseDelay

Definition at line `368` of file `include/openthread/thread_ftd.h`

otThreadGetNetworkIdTimeout

```
uint8_t otThreadGetNetworkIdTimeout (otInstance *aInstance)
```

Get the `NETWORK_ID_TIMEOUT` parameter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

Returns

- The `NETWORK_ID_TIMEOUT` value.

See Also

- otThreadSetNetworkIdTimeout

Definition at line `383` of file `include/openthread/thread_ftd.h`

otThreadSetNetworkIdTimeout

```
void otThreadSetNetworkIdTimeout (otInstance *aInstance, uint8_t aTimeout)
```

Set the `NETWORK_ID_TIMEOUT` parameter.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aTimeout	The `NETWORK_ID_TIMEOUT` value.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

See Also

- otThreadGetNetworkIdTimeout

Definition at line 397 of file include/openthread/thread_ftd.h

otThreadGetRouterUpgradeThreshold

```
uint8_t otThreadGetRouterUpgradeThreshold (otInstance *aInstance)
```

Get the ROUTER_UPGRADE_THRESHOLD parameter used in the REED role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The ROUTER_UPGRADE_THRESHOLD value.

See Also

- otThreadSetRouterUpgradeThreshold

Definition at line 409 of file include/openthread/thread_ftd.h

otThreadSetRouterUpgradeThreshold

```
void otThreadSetRouterUpgradeThreshold (otInstance *aInstance, uint8_t aThreshold)
```

Set the ROUTER_UPGRADE_THRESHOLD parameter used in the Leader role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aThreshold	The ROUTER_UPGRADE_THRESHOLD value.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

See Also

- otThreadGetRouterUpgradeThreshold

Definition at line 423 of file include/openthread/thread_ftd.h

otThreadGetChildRouterLinks

```
uint8_t otThreadGetChildRouterLinks (otInstance *aInstance)
```

Get the MLE_CHILD_ROUTER_LINKS parameter used in the REED role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

This parameter specifies the max number of neighboring routers with which the device (as an FED) will try to establish link.

Returns

- The MLE_CHILD_ROUTER_LINKS value.

See Also

- otThreadSetChildRouterLinks

Definition at line 438 of file include/openthread/thread_ftd.h

otThreadSetChildRouterLinks

```
otError otThreadSetChildRouterLinks (otInstance *aInstance, uint8_t aChildRouterLinks)
```

Set the MLE_CHILD_ROUTER_LINKS parameter used in the REED role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChildRouterLinks	The MLE_CHILD_ROUTER_LINKS value.

See Also

- otThreadGetChildRouterLinks

Definition at line 452 of file include/openthread/thread_ftd.h

otThreadReleaseRouterId

```
otError otThreadReleaseRouterId (otInstance *aInstance, uint8_t aRouterId)
```

Release a Router ID that has been allocated by the device in the Leader role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aRouterId	The Router ID to release. Valid range is [0, 62].

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

Definition at line 469 of file include/openthread/thread_ftd.h

otThreadBecomeRouter

```
otError otThreadBecomeRouter (otInstance *aInstance)
```

Attempt to become a router.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

Definition at line 482 of file include/openthread/thread_ftd.h

otThreadBecomeLeader

```
otError otThreadBecomeLeader (otInstance *aInstance)
```

Become a leader and start a new partition.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

Definition at line `495` of file `include/openthread/thread_ftd.h`

otThreadGetRouterDowngradeThreshold

```
uint8_t otThreadGetRouterDowngradeThreshold (otInstance *aInstance)
```

Get the ROUTER_DOWNGRADE_THRESHOLD parameter used in the Router role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The ROUTER_DOWNGRADE_THRESHOLD value.

See Also

- otThreadSetRouterDowngradeThreshold

Definition at line `507` of file `include/openthread/thread_ftd.h`

otThreadSetRouterDowngradeThreshold

```
void otThreadSetRouterDowngradeThreshold (otInstance *aInstance, uint8_t aThreshold)
```

Set the ROUTER_DOWNGRADE_THRESHOLD parameter used in the Leader role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aThreshold	The ROUTER_DOWNGRADE_THRESHOLD value.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

See Also

- otThreadGetRouterDowngradeThreshold

Definition at line `521` of file `include/openthread/thread_ftd.h`

otThreadGetRouterSelectionJitter

```
uint8_t otThreadGetRouterSelectionJitter (otInstance *aInstance)
```

Get the ROUTER_SELECTION_JITTER parameter used in the REED/Router role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The ROUTER_SELECTION_JITTER value.

See Also

- otThreadSetRouterSelectionJitter

Definition at line 533 of file include/openthread/thread_ftd.h

otThreadSetRouterSelectionJitter

```
void otThreadSetRouterSelectionJitter (otInstance *aInstance, uint8_t aRouterJitter)
```

Set the ROUTER_SELECTION_JITTER parameter used in the REED/Router role.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aRouterJitter	The ROUTER_SELECTION_JITTER value.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

See Also

- otThreadGetRouterSelectionJitter

Definition at line 547 of file include/openthread/thread_ftd.h

otThreadGetChildInfoById

```
otError otThreadGetChildInfoById (otInstance *aInstance, uint16_t aChildId, otChildInfo *aChildInfo)
```

Gets diagnostic information for an attached Child by its Child ID or RLOC16.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChildId	The Child ID or RLOC16 for the attached child.
[out]	aChildInfo	A pointer to where the child information is placed.

Definition at line 561 of file include/openthread/thread_ftd.h

otThreadGetChildInfoByIndex

> otError otThreadGetChildInfoByIndex (otInstance *aInstance, uint16_t aChildIndex, otChildInfo *aChildInfo)

The function retains diagnostic information for an attached Child by the internal table index.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChildIndex	The table index.
[out]	aChildInfo	A pointer to where the child information is placed.

See Also

- otGetMaxAllowedChildren

Definition at line 578 of file include/openthread/thread_ftd.h

otThreadGetChildNextIp6Address

> otError otThreadGetChildNextIp6Address (otInstance *aInstance, uint16_t aChildIndex, otChildIp6AddressIterator *aIterator, otIp6Address *aAddress)

Gets the next IPv6 address (using an iterator) for a given child.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChildIndex	The child index.
[inout]	aIterator	A pointer to the iterator. On success the iterator will be updated to point to next entry in the list. To get the first IPv6 address the iterator should be set to OT_CHILD_IP6_ADDRESS_ITERATOR_INIT.
[out]	aAddress	A pointer to an IPv6 address where the child's next address is placed (on success).

See Also

- otThreadGetChildInfoByIndex

Definition at line 597 of file include/openthread/thread_ftd.h

otThreadGetRouterIdSequence

> uint8_t otThreadGetRouterIdSequence (otInstance *aInstance)

Get the current Router ID Sequence.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Router ID Sequence.

Definition at line 610 of file include/openthread/thread_ftd.h

otThreadGetMaxRouterId

> uint8_t otThreadGetMaxRouterId (otInstance *aInstance)

The function returns the maximum allowed router ID.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The maximum allowed router ID.

Definition at line `620` of file `include/openthread/thread_ftd.h`

otThreadGetRouterInfo

> otError otThreadGetRouterInfo (otInstance *aInstance, uint16_t aRouterId, otRouterInfo *aRouterInfo)

The function retains diagnostic information for a given Thread Router.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aRouterId	The router ID or RLOC16 for a given router.
[out]	aRouterInfo	A pointer to where the router information is placed.

Definition at line `634` of file `include/openthread/thread_ftd.h`

otThreadGetNextCacheEntry

> otError otThreadGetNextCacheEntry (otInstance *aInstance, otCacheEntryInfo *aEntryInfo, otCacheEntryIterator *aIterator)

Gets the next EID cache entry (using an iterator).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aEntryInfo	A pointer to where the EID cache entry information is placed.
[inout]	aIterator	A pointer to an iterator. It will be updated to point to next entry on success. To get the first entry, initialize the iterator by setting all its fields to zero (e.g., `memset` the iterator structure to zero).

Definition at line `649` of file `include/openthread/thread_ftd.h`

otThreadGetPskc

> void otThreadGetPskc (otInstance *aInstance, otPskc *aPskc)

Get the Thread PSKc.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[out]	aPskc	A pointer to an `otPskc` to return the retrieved Thread PSKc.

See Also

- otThreadSetPskc

Definition at line `660` of file `include/openthread/thread_ftd.h`

otThreadGetPskcRef

```
otPskcRef otThreadGetPskcRef (otInstance *aInstance)
```

Get Key Reference to Thread PSKc stored.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Requires the build-time feature `OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE` to be enabled.

Returns

- Key Reference to PSKc

See Also

- otThreadSetPskcRef

Definition at line `674` of file `include/openthread/thread_ftd.h`

otThreadSetPskc

```
otError otThreadSetPskc (otInstance *aInstance, const otPskc *aPskc)
```

Set the Thread PSKc.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPskc	A pointer to the new Thread PSKc.

Will only succeed when Thread protocols are disabled. A successful call to this function will also invalidate the Active and Pending Operational Datasets in non-volatile memory.

See Also

- otThreadGetPskc

Definition at line `692` of file `include/openthread/thread_ftd.h`

otThreadSetPskcRef

```
otError otThreadSetPskcRef (otInstance *aInstance, otPskcRef aKeyRef)
```

Set the Key Reference to the Thread PSKc.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aKeyRef	Key Reference to the new Thread PSKc.

Requires the build-time feature `OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE` to be enabled.

Will only succeed when Thread protocols are disabled. Upon success, this will also invalidate the Active and Pending Operational Datasets in non-volatile memory.

See Also

- otThreadGetPskcRef

Definition at line `712` of file `include/openthread/thread_ftd.h`

otThreadGetParentPriority

```
int8_t otThreadGetParentPriority (otInstance *aInstance)
```

Get the assigned parent priority.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The assigned parent priority value, -2 means not assigned.

See Also

- otThreadSetParentPriority

Definition at line `724` of file `include/openthread/thread_ftd.h`

otThreadSetParentPriority

```
otError otThreadSetParentPriority (otInstance *aInstance, int8_t aParentPriority)
```

Set the parent priority.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aParentPriority	The parent priority value.

Note

- This API is reserved for testing and demo purposes only. Changing settings with this API will render a production application non-compliant with the Thread Specification.

See Also

- otThreadGetParentPriority

Definition at line `741` of file `include/openthread/thread_ftd.h`

otThreadGetMaxChildIpAddresses

```
uint8_t otThreadGetMaxChildIpAddresses (otInstance *aInstance)
```

Gets the maximum number of IP addresses that each MTD child may register with this device as parent.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The maximum number of IP addresses that each MTD child may register with this device as parent.

See Also

- otThreadSetMaxChildIpAddresses

Definition at line `753` of file `include/openthread/thread_ftd.h`

otThreadSetMaxChildIpAddresses

> otError otThreadSetMaxChildIpAddresses (otInstance *aInstance, uint8_t aMaxIpAddresses)

Sets or restores the maximum number of IP addresses that each MTD child may register with this device as parent.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMaxIpAddresses	The maximum number of IP addresses that each MTD child may register with this device as parent. 0 to clear the setting and restore the default.

Pass `0` to clear the setting and restore the default.

Available when `OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE` is enabled.

Note

- Only used by Thread Test Harness to limit the address registrations of the reference parent in order to test the MTD DUT reaction.

See Also

- otThreadGetMaxChildIpAddresses

Definition at line `776` of file `include/openthread/thread_ftd.h`

otThreadRegisterNeighborTableCallback

> void otThreadRegisterNeighborTableCallback (otInstance *aInstance, otNeighborTableCallback aCallback)

Registers a neighbor table callback function.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to callback handler function.

The provided callback (if non-NULL) will be invoked when there is a change in the neighbor table (e.g., a child or a router neighbor entry is being added/removed or an existing child's mode is changed).

Subsequent calls to this method will overwrite the previous callback. Note that this callback in invoked while the neighbor/child table is being updated and always before the `otStateChangedCallback`.

Definition at line 828 of file include/openthread/thread_ftd.h

otThreadSetCcmEnabled

> void otThreadSetCcmEnabled (otInstance *aInstance, bool aEnabled)

Sets whether the device was commissioned using CCM.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	TRUE if the device was commissioned using CCM, FALSE otherwise.

Note

- This API requires OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE, and is only used by Thread Test Harness to indicate whether this device was commissioned using CCM.

Definition at line 840 of file include/openthread/thread_ftd.h

otThreadSetThreadVersionCheckEnabled

> void otThreadSetThreadVersionCheckEnabled (otInstance *aInstance, bool aEnabled)

Sets whether the Security Policy TLV version-threshold for routing (VR field) is enabled.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	TRUE to enable Security Policy TLV version-threshold for routing, FALSE otherwise.

Note

- This API requires OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE, and is only used by Thread Test Harness to indicate that thread protocol version check VR should be skipped.

Definition at line 852 of file include/openthread/thread_ftd.h

otThreadGetRouterIdRange

> void otThreadGetRouterIdRange (otInstance *aInstance, uint8_t *aMinRouterId, uint8_t *aMaxRouterId)

Gets the range of router IDs that are allowed to assign to nodes within the thread network.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[out]	aMinRouterId	The minimum router ID.
[out]	aMaxRouterId	The maximum router ID.

Note

- This API requires OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE, and is only used for test purpose. All the router IDs in the range [aMinRouterId, aMaxRouterId] are allowed.

See Also

- otThreadSetRouterIdRange

Definition at line 867 of file include/openthread/thread_ftd.h

otThreadSetRouterIdRange

```
otError otThreadSetRouterIdRange (otInstance *aInstance, uint8_t aMinRouterId, uint8_t aMaxRouterId)
```

Sets the range of router IDs that are allowed to assign to nodes within the thread network.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMinRouterId	The minimum router ID.
[in]	aMaxRouterId	The maximum router ID.

Note

- This API requires OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE, and is only used for test purpose. All the router IDs in the range [aMinRouterId, aMaxRouterId] are allowed.

See Also

- otThreadGetRouterIdRange

Definition at line 885 of file include/openthread/thread_ftd.h

otThreadGetAdvertisementTrickleIntervalMax

```
uint32_t otThreadGetAdvertisementTrickleIntervalMax (otInstance *aInstance)
```

Gets the current Interval Max value used by Advertisement trickle timer.

Parameters

N/A	aInstance	

This API requires OPENTHREAD_CONFIG_REFERENCE_DEVICE_ENABLE, and is intended for testing only.

Returns

- The Interval Max of Advertisement trickle timer in milliseconds.

Definition at line 895 of file include/openthread/thread_ftd.h

otThreadIsRouterIdAllocated

```
bool otThreadIsRouterIdAllocated (otInstance *aInstance, uint8_t aRouterId)
```

Indicates whether or not a Router ID is currently allocated.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aRouterId	The router ID to check.

Definition at line 907 of file include/openthread/thread_ftd.h

otThreadGetNextHopAndPathCost

```
void otThreadGetNextHopAndPathCost (otInstance *aInstance, uint16_t aDestRloc16, uint16_t *aNextHopRloc16, uint8_t *aPathCost)
```

Gets the next hop and path cost towards a given RLOC16 destination.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDestRloc16	The RLOC16 of destination.
[out]	aNextHopRloc16	A pointer to return RLOC16 of next hop, 0xfffe if no next hop.
[out]	aPathCost	A pointer to return path cost towards destination.

Can be used with either `aNextHopRloc16` or `aPathCost` being NULL indicating caller does not want to get the value.

Definition at line `921` of file `include/openthread/thread_ftd.h`

Macro Definition Documentation

OT_CHILD_IP6_ADDRESS_ITERATOR_INIT

```
#define OT_CHILD_IP6_ADDRESS_ITERATOR_INIT
```

Value:

```
0
```

Initializer for otChildIP6AddressIterator.

Definition at line `81` of file `include/openthread/thread_ftd.h`

otChildInfo

Holds diagnostic information for a Thread Child.

Public Attributes

otExtAddress	mExtAddress IEEE 802.15.4 Extended Address.
uint32_t	mTimeout Timeout.
uint32_t	mAge Seconds since last heard.
uint64_t	mConnectionTime Seconds since attach (requires `OPENTHREAD_CONFIG_UPTIME_ENABLE`)
uint16_t	mRloc16 RLOC16.
uint16_t	mChildId Child ID.
uint8_t	mNetworkDataVersion Network Data Version.
uint8_t	mLinkQualityIn Link Quality In.
int8_t	mAverageRssi Average RSSI.
int8_t	mLastRssi Last observed RSSI.
uint16_t	mFrameErrorRate Frame error rate (0xffff->100%). Requires error tracking feature.
uint16_t	mMessageErrorRate (IPv6) msg error rate (0xffff->100%). Requires error tracking feature.
uint16_t	mQueuedMessageCnt Number of queued messages for the child.
uint16_t	mSupervisionInterval Supervision interval (in seconds).
uint8_t	mVersion MLE version.
bool	mRxOnWhenIdle rx-on-when-idle
bool	mFullThreadDevice Full Thread Device.
bool	mFullNetworkData Full Network Data.

bool mIsStateRestoring
Is in restoring state.

bool mIsCslSynced
Is child CSL synchronized.

Public Attribute Documentation

mExtAddress

```
otExtAddress otChildInfo::mExtAddress
```

IEEE 802.15.4 Extended Address.

Definition at line 59 of file include/openthread/thread_ftd.h

mTimeout

```
uint32_t otChildInfo::mTimeout
```

Timeout.

Definition at line 60 of file include/openthread/thread_ftd.h

mAge

```
uint32_t otChildInfo::mAge
```

Seconds since last heard.

Definition at line 61 of file include/openthread/thread_ftd.h

mConnectionTime

```
uint64_t otChildInfo::mConnectionTime
```

Seconds since attach (requires OPENTHREAD_CONFIG_UPTIME_ENABLE)

Definition at line 62 of file include/openthread/thread_ftd.h

mRloc16

```
uint16_t otChildInfo::mRloc16
```

RLOC16.

Definition at line 63 of file include/openthread/thread_ftd.h

mChildId

uint16_t otChildInfo::mChildId

Child ID.

Definition at line 64 of file include/openthread/thread_ftd.h

mNetworkDataVersion

uint8_t otChildInfo::mNetworkDataVersion

Network Data Version.

Definition at line 65 of file include/openthread/thread_ftd.h

mLinkQualityIn

uint8_t otChildInfo::mLinkQualityIn

Link Quality In.

Definition at line 66 of file include/openthread/thread_ftd.h

mAverageRssi

int8_t otChildInfo::mAverageRssi

Average RSSI.

Definition at line 67 of file include/openthread/thread_ftd.h

mLastRssi

int8_t otChildInfo::mLastRssi

Last observed RSSI.

Definition at line 68 of file include/openthread/thread_ftd.h

mFrameErrorRate

uint16_t otChildInfo::mFrameErrorRate

Frame error rate (0xffff->100%). Requires error tracking feature.

Definition at line 69 of file include/openthread/thread_ftd.h

mMessageErrorRate

```
uint16_t otChildInfo::mMessageErrorRate
```

(IPv6) msg error rate (0xffff->100%). Requires error tracking feature.

Definition at line `70` of file `include/openthread/thread_ftd.h`

mQueuedMessageCnt

```
uint16_t otChildInfo::mQueuedMessageCnt
```

Number of queued messages for the child.

Definition at line `71` of file `include/openthread/thread_ftd.h`

mSupervisionInterval

```
uint16_t otChildInfo::mSupervisionInterval
```

Supervision interval (in seconds).

Definition at line `72` of file `include/openthread/thread_ftd.h`

mVersion

```
uint8_t otChildInfo::mVersion
```

MLE version.

Definition at line `73` of file `include/openthread/thread_ftd.h`

mRxOnWhenIdle

```
bool otChildInfo::mRxOnWhenIdle
```

rx-on-when-idle

Definition at line `74` of file `include/openthread/thread_ftd.h`

mFullThreadDevice

```
bool otChildInfo::mFullThreadDevice
```

Full Thread Device.

Definition at line `75` of file `include/openthread/thread_ftd.h`

mFullNetworkData

bool otChildInfo::mFullNetworkData

Full Network Data.

Definition at line 76 of file include/openthread/thread_ftd.h

mIsStateRestoring

bool otChildInfo::mIsStateRestoring

Is in restoring state.

Definition at line 77 of file include/openthread/thread_ftd.h

mIsCslSynced

bool otChildInfo::mIsCslSynced

Is child CSL synchronized.

Definition at line 78 of file include/openthread/thread_ftd.h

otCacheEntryInfo

Represents an EID cache entry.

Public Attributes

otIp6Address	mTarget Target EID.
otShortAddress	mRloc16 RLOC16.
otCacheEntryState	mState Entry state.
bool	mCanEvict Indicates whether the entry can be evicted.
bool	mRampDown Whether in ramp-down mode while in `OT_CACHE_ENTRY_STATE_RETRY_QUERY`.
bool	mValidLastTrans Indicates whether last transaction time and ML-EID are valid.
uint32_t	mLastTransTime Last transaction time (applicable in cached state).
otIp6Address	mMeshLocalEid Mesh Local EID (applicable if entry in cached state).
uint16_t	mTimeout Timeout in seconds (applicable if in snooped/query/retry-query states).
uint16_t	mRetryDelay Retry delay in seconds (applicable if in query-retry state).

Public Attribute Documentation

mTarget

> otIp6Address otCacheEntryInfo::mTarget

Target EID.

Definition at line `103` of file `include/openthread/thread_ftd.h`

mRloc16

> otShortAddress otCacheEntryInfo::mRloc16

RLOC16.

Definition at line `104` of file `include/openthread/thread_ftd.h`

mState

```
otCacheEntryState otCacheEntryInfo::mState
```

Entry state.

Definition at line 105 of file include/openthread/thread_ftd.h

mCanEvict

```
bool otCacheEntryInfo::mCanEvict
```

Indicates whether the entry can be evicted.

Definition at line 106 of file include/openthread/thread_ftd.h

mRampDown

```
bool otCacheEntryInfo::mRampDown
```

Whether in ramp-down mode while in OT_CACHE_ENTRY_STATE_RETRY_QUERY.

Definition at line 107 of file include/openthread/thread_ftd.h

mValidLastTrans

```
bool otCacheEntryInfo::mValidLastTrans
```

Indicates whether last transaction time and ML-EID are valid.

Definition at line 108 of file include/openthread/thread_ftd.h

mLastTransTime

```
uint32_t otCacheEntryInfo::mLastTransTime
```

Last transaction time (applicable in cached state).

Definition at line 109 of file include/openthread/thread_ftd.h

mMeshLocalEid

```
otIp6Address otCacheEntryInfo::mMeshLocalEid
```

Mesh Local EID (applicable if entry in cached state).

Definition at line 110 of file include/openthread/thread_ftd.h

mTimeout

> uint16_t otCacheEntryInfo::mTimeout

Timeout in seconds (applicable if in snooped/query/retry-query states).

Definition at line 111 of file include/openthread/thread_ftd.h

mRetryDelay

> uint16_t otCacheEntryInfo::mRetryDelay

Retry delay in seconds (applicable if in query-retry state).

Definition at line 112 of file include/openthread/thread_ftd.h

otCacheEntryIterator

Represents an iterator used for iterating through the EID cache table entries.

To initialize the iterator and start from the first entry in the cache table, set all its fields in the structure to zero (e.g., `memset` the iterator to zero).

Public Attributes

const void *	mData Opaque data used by the core implementation. Should not be changed by user.

Public Attribute Documentation

mData

```
const void* otCacheEntryIterator::mData[2]
```

Opaque data used by the core implementation. Should not be changed by user.

Definition at line `124` of file `include/openthread/thread_ftd.h`

otDeviceProperties

Represents the device properties which are used for calculating the local leader weight on a device.

The parameters are set based on device's capability, whether acting as border router, its power supply config, etc.

`mIsUnstable` indicates operational stability of device and is determined via a vendor specific mechanism. It can include the following cases:

- Device internally detects that it loses external power supply more often than usual. What is usual is determined by the vendor.
- Device internally detects that it reboots more often than usual. What is usual is determined by the vendor.

Public Attributes

otPowerSupply	mPowerSupply Power supply config.
bool	mIsBorderRouter Whether device is a border router.
bool	mSupportsCcm Whether device supports CCM (can act as a CCM border router).
bool	mIsUnstable Operational stability of device (vendor specific).
int8_t	mLeaderWeightAdjustment Weight adjustment. Should be -16 to +16 (clamped otherwise).

Public Attribute Documentation

mPowerSupply

> otPowerSupply otDeviceProperties::mPowerSupply

Power supply config.

Definition at line `230` of file `include/openthread/thread_ftd.h`

mIsBorderRouter

> bool otDeviceProperties::mIsBorderRouter

Whether device is a border router.

Definition at line `231` of file `include/openthread/thread_ftd.h`

mSupportsCcm

bool otDeviceProperties::mSupportsCcm

Whether device supports CCM (can act as a CCM border router).

Definition at line 232 of file include/openthread/thread_ftd.h

mIsUnstable

bool otDeviceProperties::mIsUnstable

Operational stability of device (vendor specific).

Definition at line 233 of file include/openthread/thread_ftd.h

mLeaderWeightAdjustment

int8_t otDeviceProperties::mLeaderWeightAdjustment

Weight adjustment. Should be -16 to +16 (clamped otherwise).

Definition at line 234 of file include/openthread/thread_ftd.h

otNeighborTableEntryInfo

Represent a neighbor table entry info (child or router) and is used as a parameter in the neighbor table callback `otNeighborTableCallback`.

Public Attributes

otInstance *	mInstance The OpenThread instance.
otChildInfo	mChild The child neighbor info.
otNeighborInfo	mRouter The router neighbor info.
union otNeighborTableEntryInfo::@27	mInfo

Public Attribute Documentation

mInstance

```
otInstance* otNeighborTableEntryInfo::mInstance
```

The OpenThread instance.

Definition at line `798` of file `include/openthread/thread_ftd.h`

mChild

```
otChildInfo otNeighborTableEntryInfo::mChild
```

The child neighbor info.

Definition at line `801` of file `include/openthread/thread_ftd.h`

mRouter

```
otNeighborInfo otNeighborTableEntryInfo::mRouter
```

The router neighbor info.

Definition at line `802` of file `include/openthread/thread_ftd.h`

mInfo

```
union otNeighborTableEntryInfo::@27 otNeighborTableEntryInfo::mInfo
```

Definition at line 803 of file include/openthread/thread_ftd.h

Server

Server

This module includes functions to manage local network data with the OpenThread Server.

Functions

otError otServerGetNetDataLocal(otInstance *aInstance, bool aStable, uint8_t *aData, uint8_t *aDataLength)
Provides a full or stable copy of the local Thread Network Data.

otError otServerAddService(otInstance *aInstance, const otServiceConfig *aConfig)
Add a service configuration to the local network data.

otError otServerRemoveService(otInstance *aInstance, uint32_t aEnterpriseNumber, const uint8_t *aServiceData, uint8_t aServiceDataLength)
Remove a service configuration from the local network data.

otError otServerGetNextService(otInstance *aInstance, otNetworkDataIterator *aIterator, otServiceConfig *aConfig)
Gets the next service in the local Network Data.

otError otServerRegister(otInstance *aInstance)
Immediately register the local network data with the Leader.

Function Documentation

otServerGetNetDataLocal

> otError otServerGetNetDataLocal (otInstance *aInstance, bool aStable, uint8_t *aData, uint8_t *aDataLength)

Provides a full or stable copy of the local Thread Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aStable	TRUE when copying the stable version, FALSE when copying the full version.
[out]	aData	A pointer to the data buffer.
[inout]	aDataLength	On entry, size of the data buffer pointed to by `aData`. On exit, number of copied bytes.

Definition at line `64` of file `include/openthread/server.h`

otServerAddService

> otError otServerAddService (otInstance *aInstance, const otServiceConfig *aConfig)

Add a service configuration to the local network data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[in]	aConfig	A pointer to the service configuration.

See Also

- otServerRemoveService
- otServerRegister

Definition at line 80 of file include/openthread/server.h

otServerRemoveService

> otError otServerRemoveService (otInstance *aInstance, uint32_t aEnterpriseNumber, const uint8_t *aServiceData, uint8_t aServiceDataLength)

Remove a service configuration from the local network data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnterpriseNumber	Enterprise Number of the service entry to be deleted.
[in]	aServiceData	A pointer to an Service Data to look for during deletion.
[in]	aServiceDataLength	The length of aServiceData in bytes.

See Also

- otServerAddService
- otServerRegister

Definition at line 97 of file include/openthread/server.h

otServerGetNextService

> otError otServerGetNextService (otInstance *aInstance, otNetworkDataIterator *aIterator, otServiceConfig *aConfig)

Gets the next service in the local Network Data.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[inout]	aIterator	A pointer to the Network Data iterator context. To get the first service entry it should be set to OT_NETWORK_DATA_ITERATOR_INIT.
[out]	aConfig	A pointer to where the service information will be placed.

Definition at line 114 of file include/openthread/server.h

otServerRegister

> otError otServerRegister (otInstance *aInstance)

Immediately register the local network data with the Leader.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

See Also

- otServerAddService
- otServerRemoveService

Definition at line 127 of file include/openthread/server.h

Add-Ons

Add-Ons

Modules

- Channel Manager
- Channel Monitoring
- Child Supervision
- CoAP
- Command Line Interface
- Crypto - Thread Stack
- Factory Diagnostics - Thread Stack
- Heap
- History Tracker
- Jam Detection
- Logging - Thread Stack
- Mesh Diagnostics
- Network Co-Processor
- Network Time Synchronization
- Radio Statistics
- Random Number Generator
- SNTP

Channel Manager

Channel Manager

This module includes functions for Channel Manager.

The functions in this module are available when Channel Manager feature (`OPENTHREAD_CONFIG_CHANNEL_MANAGER_ENABLE`) is enabled. Channel Manager is available only on an FTD build.

Functions

void otChannelManagerRequestChannelChange(otInstance *aInstance, uint8_t aChannel)
Requests a Thread network channel change.

uint8_t otChannelManagerGetRequestedChannel(otInstance *aInstance)
Gets the channel from the last successful call to `otChannelManagerRequestChannelChange()`

uint16_t otChannelManagerGetDelay(otInstance *aInstance)
Gets the delay (in seconds) used by Channel Manager for a channel change.

otError otChannelManagerSetDelay(otInstance *aInstance, uint16_t aDelay)
Sets the delay (in seconds) used for a channel change.

otError otChannelManagerRequestChannelSelect(otInstance *aInstance, bool aSkipQualityCheck)
Requests that `ChannelManager` checks and selects a new channel and starts a channel change.

void otChannelManagerSetAutoChannelSelectionEnabled(otInstance *aInstance, bool aEnabled)
Enables or disables the auto-channel-selection functionality.

bool otChannelManagerGetAutoChannelSelectionEnabled(otInstance *aInstance)
Indicates whether the auto-channel-selection functionality is enabled or not.

otError otChannelManagerSetAutoChannelSelectionInterval(otInstance *aInstance, uint32_t aInterval)
Sets the period interval (in seconds) used by auto-channel-selection functionality.

uint32_t otChannelManagerGetAutoChannelSelectionInterval(otInstance *aInstance)
Gets the period interval (in seconds) used by auto-channel-selection functionality.

uint32_t otChannelManagerGetSupportedChannels(otInstance *aInstance)
Gets the supported channel mask.

void otChannelManagerSetSupportedChannels(otInstance *aInstance, uint32_t aChannelMask)
Sets the supported channel mask.

uint32_t otChannelManagerGetFavoredChannels(otInstance *aInstance)
Gets the favored channel mask.

void otChannelManagerSetFavoredChannels(otInstance *aInstance, uint32_t aChannelMask)
Sets the favored channel mask.

uint16_t otChannelManagerGetCcaFailureRateThreshold(otInstance *aInstance)
Gets the CCA failure rate threshold.

void otChannelManagerSetCcaFailureRateThreshold(otInstance *aInstance, uint16_t aThreshold)
Sets the CCA failure rate threshold.

Function Documentation

otChannelManagerRequestChannelChange

```
void otChannelManagerRequestChannelChange (otInstance *aInstance, uint8_t aChannel)
```

Requests a Thread network channel change.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChannel	The new channel for the Thread network.

The network switches to the given channel after a specified delay (see otChannelManagerSetDelay()). The channel change is performed by updating the Pending Operational Dataset.

A subsequent call will cancel an ongoing previously requested channel change.

Definition at line `69` of file `include/openthread/channel_manager.h`

otChannelManagerGetRequestedChannel

```
uint8_t otChannelManagerGetRequestedChannel (otInstance *aInstance)
```

Gets the channel from the last successful call to `otChannelManagerRequestChannelChange()`

Parameters

N/A	aInstance	

Returns

- The last requested channel or zero if there has been no channel change request yet.

Definition at line `77` of file `include/openthread/channel_manager.h`

otChannelManagerGetDelay

```
uint16_t otChannelManagerGetDelay (otInstance *aInstance)
```

Gets the delay (in seconds) used by Channel Manager for a channel change.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The delay (in seconds) for channel change.

Definition at line `87` of file `include/openthread/channel_manager.h`

otChannelManagerSetDelay

```
otError otChannelManagerSetDelay (otInstance *aInstance, uint16_t aDelay)
```

Sets the delay (in seconds) used for a channel change.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aDelay	Delay in seconds.

The delay should preferably be longer than the maximum data poll interval used by all sleepy-end-devices within the Thread network.

Definition at line `102` of file `include/openthread/channel_manager.h`

otChannelManagerRequestChannelSelect

```
otError otChannelManagerRequestChannelSelect (otInstance *aInstance, bool aSkipQualityCheck)
```

Requests that `ChannelManager` checks and selects a new channel and starts a channel change.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSkipQualityCheck	Indicates whether the quality check (step 1) should be skipped.

Unlike the `otChannelManagerRequestChannelChange()` where the channel must be given as a parameter, this function asks the `ChannelManager` to select a channel by itself (based on collected channel quality info).

Once called, the Channel Manager will perform the following 3 steps:

1) `ChannelManager` decides if the channel change would be helpful. This check can be skipped if `aSkipQualityCheck` is set to true (forcing a channel selection to happen and skipping the quality check). This step uses the collected link quality metrics on the device (such as CCA failure rate, frame and message error rates per neighbor, etc.) to determine if the current channel quality is at the level that justifies a channel change.

2) If the first step passes, then `ChannelManager` selects a potentially better channel. It uses the collected channel quality data by `ChannelMonitor` module. The supported and favored channels are used at this step. (see otChannelManagerSetSupportedChannels() and otChannelManagerSetFavoredChannels()).

3) If the newly selected channel is different from the current channel, `ChannelManager` requests/starts the channel change process (internally invoking a `RequestChannelChange()`).

Definition at line `132` of file `include/openthread/channel_manager.h`

otChannelManagerSetAutoChannelSelectionEnabled

```
void otChannelManagerSetAutoChannelSelectionEnabled (otInstance *aInstance, bool aEnabled)
```

Enables or disables the auto-channel-selection functionality.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	Indicates whether to enable or disable this functionality.

When enabled, `ChannelManager` will periodically invoke a `RequestChannelSelect(false)` . The period interval can be set by `SetAutoChannelSelectionInterval()` .

Definition at line `144` of file `include/openthread/channel_manager.h`

otChannelManagerGetAutoChannelSelectionEnabled

```
bool otChannelManagerGetAutoChannelSelectionEnabled (otInstance *aInstance)
```

Indicates whether the auto-channel-selection functionality is enabled or not.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- TRUE if enabled, FALSE if disabled.

Definition at line `154` of file `include/openthread/channel_manager.h`

otChannelManagerSetAutoChannelSelectionInterval

```
otError otChannelManagerSetAutoChannelSelectionInterval (otInstance *aInstance, uint32_t aInterval)
```

Sets the period interval (in seconds) used by auto-channel-selection functionality.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aInterval	The interval in seconds.

Definition at line `166` of file `include/openthread/channel_manager.h`

otChannelManagerGetAutoChannelSelectionInterval

```
uint32_t otChannelManagerGetAutoChannelSelectionInterval (otInstance *aInstance)
```

Gets the period interval (in seconds) used by auto-channel-selection functionality.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The interval in seconds.

Definition at line `176` of file `include/openthread/channel_manager.h`

otChannelManagerGetSupportedChannels

```
uint32_t otChannelManagerGetSupportedChannels (otInstance *aInstance)
```

Gets the supported channel mask.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

The supported channels as a bit-mask.

Definition at line 186 of file include/openthread/channel_manager.h

otChannelManagerSetSupportedChannels

```
void otChannelManagerSetSupportedChannels (otInstance *aInstance, uint32_t aChannelMask)
```

Sets the supported channel mask.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChannelMask	A channel mask.

Definition at line 195 of file include/openthread/channel_manager.h

otChannelManagerGetFavoredChannels

```
uint32_t otChannelManagerGetFavoredChannels (otInstance *aInstance)
```

Gets the favored channel mask.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The favored channels as a bit-mask.

Definition at line 205 of file include/openthread/channel_manager.h

otChannelManagerSetFavoredChannels

```
void otChannelManagerSetFavoredChannels (otInstance *aInstance, uint32_t aChannelMask)
```

Sets the favored channel mask.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChannelMask	A channel mask.

Definition at line 214 of file include/openthread/channel_manager.h

otChannelManagerGetCcaFailureRateThreshold

```
uint16_t otChannelManagerGetCcaFailureRateThreshold (otInstance *aInstance)
```

Gets the CCA failure rate threshold.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The CCA failure rate threshold. Value 0 maps to 0% and 0xffff maps to 100%.

Definition at line 224 of file include/openthread/channel_manager.h

otChannelManagerSetCcaFailureRateThreshold

```
void otChannelManagerSetCcaFailureRateThreshold (otInstance *aInstance, uint16_t aThreshold)
```

Sets the CCA failure rate threshold.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aThreshold	A CCA failure rate threshold. Value 0 maps to 0% and 0xffff maps to 100%.

Definition at line 233 of file include/openthread/channel_manager.h

Channel Monitoring

Channel Monitoring

This module includes functions for channel monitoring feature.

The functions in this module are available when channel monitor feature (`OPENTHREAD_CONFIG_CHANNEL_MONITOR_ENABLE`) is enabled.

Channel monitoring will periodically monitor all channels to help determine the cleaner channels (channels with less interference).

When channel monitoring is active, a zero-duration Energy Scan is performed, collecting a single RSSI sample on every channel per sample interval. The RSSI samples are compared with a pre-specified RSSI threshold. As an indicator of channel quality, the channel monitoring module maintains and provides the average rate/percentage of RSSI samples that are above the threshold within (approximately) a specified sample window (referred to as channel occupancy).

Functions

otError	otChannelMonitorSetEnabled(otInstance *aInstance, bool aEnabled) Enables or disables the Channel Monitoring operation.
bool	otChannelMonitorIsEnabled(otInstance *aInstance) Indicates whether the Channel Monitoring operation is enabled and running.
uint32_t	otChannelMonitorGetSampleInterval(otInstance *aInstance) Get channel monitoring sample interval in milliseconds.
int8_t	otChannelMonitorGetRssiThreshold(otInstance *aInstance) Get channel monitoring RSSI threshold in dBm.
uint32_t	otChannelMonitorGetSampleWindow(otInstance *aInstance) Get channel monitoring averaging sample window length (number of samples).
uint32_t	otChannelMonitorGetSampleCount(otInstance *aInstance) Get channel monitoring total number of RSSI samples (per channel).
uint16_t	otChannelMonitorGetChannelOccupancy(otInstance *aInstance, uint8_t aChannel) Gets the current channel occupancy for a given channel.

Function Documentation

otChannelMonitorSetEnabled

> otError otChannelMonitorSetEnabled (otInstance *aInstance, bool aEnabled)

Enables or disables the Channel Monitoring operation.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aEnabled	TRUE to enable/start Channel Monitoring operation, FALSE to disable/stop it.

Once operation starts, any previously collected data is cleared. However, after operation is disabled, the previous collected data is still valid and can be read.

Note

- OpenThread core internally enables or disables the Channel Monitoring operation when the IPv6 interface is brought up or down, for example in a call to `otIp6SetEnabled()`.

Definition at line `82` of file `include/openthread/channel_monitor.h`

otChannelMonitorIsEnabled

```
bool otChannelMonitorIsEnabled (otInstance *aInstance)
```

Indicates whether the Channel Monitoring operation is enabled and running.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- TRUE if the Channel Monitoring operation is enabled, FALSE otherwise.

Definition at line `92` of file `include/openthread/channel_monitor.h`

otChannelMonitorGetSampleInterval

```
uint32_t otChannelMonitorGetSampleInterval (otInstance *aInstance)
```

Get channel monitoring sample interval in milliseconds.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The channel monitor sample interval in milliseconds.

Definition at line `102` of file `include/openthread/channel_monitor.h`

otChannelMonitorGetRssiThreshold

```
int8_t otChannelMonitorGetRssiThreshold (otInstance *aInstance)
```

Get channel monitoring RSSI threshold in dBm.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The RSSI threshold in dBm.

Definition at line `112` of file `include/openthread/channel_monitor.h`

otChannelMonitorGetSampleWindow

```
uint32_t otChannelMonitorGetSampleWindow (otInstance *aInstance)
```

Get channel monitoring averaging sample window length (number of samples).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The averaging sample window.

Definition at line `122` of file `include/openthread/channel_monitor.h`

otChannelMonitorGetSampleCount

```
uint32_t otChannelMonitorGetSampleCount (otInstance *aInstance)
```

Get channel monitoring total number of RSSI samples (per channel).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The count indicates total number samples per channel by channel monitoring module since its start (since Thread network interface was enabled).

Returns

- Total number of RSSI samples (per channel) taken so far.

Definition at line `135` of file `include/openthread/channel_monitor.h`

otChannelMonitorGetChannelOccupancy

```
uint16_t otChannelMonitorGetChannelOccupancy (otInstance *aInstance, uint8_t aChannel)
```

Gets the current channel occupancy for a given channel.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aChannel	The channel for which to get the link occupancy.

The channel occupancy value represents the average rate/percentage of RSSI samples that were above RSSI threshold ("bad" RSSI samples).

For the first "sample window" samples, the average is maintained as the actual percentage (i.e., ratio of number of "bad" samples by total number of samples). After "window" samples, the averager uses an exponentially weighted moving average. Practically, this means the average is representative of up to `3 * window` last samples with highest weight given to the latest `kSampleWindow` samples.

Max value of `0xffff` indicates all RSSI samples were above RSSI threshold (i.e. 100% of samples were "bad").

Returns

- The current channel occupancy for the given channel.

Definition at line 156 of file include/openthread/channel_monitor.h

Child Supervision

Child Supervision

This module includes functions for Child Supervision feature.

Functions

uint16_t otChildSupervisionGetInterval(otInstance *aInstance)
Gets the Child Supervision interval (in seconds) on a child.

void otChildSupervisionSetInterval(otInstance *aInstance, uint16_t aInterval)
Sets the child supervision interval (in seconds) on the child.

uint16_t otChildSupervisionGetCheckTimeout(otInstance *aInstance)
Gets the supervision check timeout interval (in seconds) on the child.

void otChildSupervisionSetCheckTimeout(otInstance *aInstance, uint16_t aTimeout)
Sets the supervision check timeout interval (in seconds) on the child.

uint16_t otChildSupervisionGetCheckFailureCounter(otInstance *aInstance)
Get the value of supervision check timeout failure counter.

void otChildSupervisionResetCheckFailureCounter(otInstance *aInstance)
Reset the supervision check timeout failure counter to zero.

Function Documentation

otChildSupervisionGetInterval

```
uint16_t otChildSupervisionGetInterval (otInstance *aInstance)
```

Gets the Child Supervision interval (in seconds) on a child.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Child Supervision feature provides a mechanism for parent to ensure that a message is sent to each sleepy child within the supervision interval. If there is no transmission to the child within the supervision interval, OpenThread enqueues and sends a Child Supervision Message to the child.

Returns

- The child supervision interval. Zero indicates that supervision is disabled.

Definition at line 66 of file include/openthread/child_supervision.h

otChildSupervisionSetInterval

```
void otChildSupervisionSetInterval (otInstance *aInstance, uint16_t aInterval)
```

Sets the child supervision interval (in seconds) on the child.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aInterval	The supervision interval (in seconds). Zero to disable supervision.

Definition at line 75 of file include/openthread/child_supervision.h

otChildSupervisionGetCheckTimeout

```
uint16_t otChildSupervisionGetCheckTimeout (otInstance *aInstance)
```

Gets the supervision check timeout interval (in seconds) on the child.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

If the device is a sleepy child and it does not hear from its parent within the specified check timeout, it initiates the re-attach process (MLE Child Update Request/Response exchange with its parent).

Returns

- The supervision check timeout. Zero indicates that supervision check on the child is disabled.

Definition at line 88 of file include/openthread/child_supervision.h

otChildSupervisionSetCheckTimeout

```
void otChildSupervisionSetCheckTimeout (otInstance *aInstance, uint16_t aTimeout)
```

Sets the supervision check timeout interval (in seconds) on the child.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aTimeout	The check timeout (in seconds). Zero to disable supervision check on the child.

Definition at line 97 of file include/openthread/child_supervision.h

otChildSupervisionGetCheckFailureCounter

```
uint16_t otChildSupervisionGetCheckFailureCounter (otInstance *aInstance)
```

Get the value of supervision check timeout failure counter.

Parameters

N/A	aInstance	

The counter tracks the number of supervision check failures on the child. It is incremented when the child does not hear from its parent within the specified check timeout interval.

Definition at line 106 of file include/openthread/child_supervision.h

otChildSupervisionResetCheckFailureCounter

```
void otChildSupervisionResetCheckFailureCounter (otInstance *aInstance)
```

Reset the supervision check timeout failure counter to zero.

Parameters

N/A	aInstance	

Definition at line 112 of file include/openthread/child_supervision.h

CoAP

CoAP

Modules

- CoAP
- CoAP Secure

CoAP

CoAP

This module includes functions that control CoAP communication.

The functions in this module are available when CoAP API feature (`OPENTHREAD_CONFIG_COAP_API_ENABLE`) is enabled.

Modules

otCoapOption

otCoapOptionIterator

otCoapResource

otCoapBlockwiseResource

otCoapTxParameters

Enumerations

enum otCoapType {

OT_COAP_TYPE_CONFIRMABLE = 0
OT_COAP_TYPE_NON_CONFIRMABLE = 1
OT_COAP_TYPE_ACKNOWLEDGMENT = 2
OT_COAP_TYPE_RESET = 3

}

CoAP Type values (2 bit unsigned integer).

enum otCoapCode {

OT_COAP_CODE_EMPTY = OT_COAP_CODE(0, 0)
OT_COAP_CODE_GET = OT_COAP_CODE(0, 1)
OT_COAP_CODE_POST = OT_COAP_CODE(0, 2)
OT_COAP_CODE_PUT = OT_COAP_CODE(0, 3)
OT_COAP_CODE_DELETE = OT_COAP_CODE(0, 4)
OT_COAP_CODE_RESPONSE_MIN = OT_COAP_CODE(2, 0)
OT_COAP_CODE_CREATED = OT_COAP_CODE(2, 1)
OT_COAP_CODE_DELETED = OT_COAP_CODE(2, 2)
OT_COAP_CODE_VALID = OT_COAP_CODE(2, 3)
OT_COAP_CODE_CHANGED = OT_COAP_CODE(2, 4)
OT_COAP_CODE_CONTENT = OT_COAP_CODE(2, 5)
OT_COAP_CODE_CONTINUE = OT_COAP_CODE(2, 31)
OT_COAP_CODE_BAD_REQUEST = OT_COAP_CODE(4, 0)
OT_COAP_CODE_UNAUTHORIZED = OT_COAP_CODE(4, 1)
OT_COAP_CODE_BAD_OPTION = OT_COAP_CODE(4, 2)
OT_COAP_CODE_FORBIDDEN = OT_COAP_CODE(4, 3)
OT_COAP_CODE_NOT_FOUND = OT_COAP_CODE(4, 4)
OT_COAP_CODE_METHOD_NOT_ALLOWED = OT_COAP_CODE(4, 5)
OT_COAP_CODE_NOT_ACCEPTABLE = OT_COAP_CODE(4, 6)
OT_COAP_CODE_REQUEST_INCOMPLETE = OT_COAP_CODE(4, 8)
OT_COAP_CODE_PRECONDITION_FAILED = OT_COAP_CODE(4, 12)
OT_COAP_CODE_REQUEST_TOO_LARGE = OT_COAP_CODE(4, 13)
OT_COAP_CODE_UNSUPPORTED_FORMAT = OT_COAP_CODE(4, 15)
OT_COAP_CODE_INTERNAL_ERROR = OT_COAP_CODE(5, 0)
OT_COAP_CODE_NOT_IMPLEMENTED = OT_COAP_CODE(5, 1)
OT_COAP_CODE_BAD_GATEWAY = OT_COAP_CODE(5, 2)
OT_COAP_CODE_SERVICE_UNAVAILABLE = OT_COAP_CODE(5, 3)
OT_COAP_CODE_GATEWAY_TIMEOUT = OT_COAP_CODE(5, 4)
OT_COAP_CODE_PROXY_NOT_SUPPORTED = OT_COAP_CODE(5, 5)

}

CoAP Code values.

enum otCoapOptionType {

OT_COAP_OPTION_IF_MATCH = 1
OT_COAP_OPTION_URI_HOST = 3
OT_COAP_OPTION_E_TAG = 4
OT_COAP_OPTION_IF_NONE_MATCH = 5
OT_COAP_OPTION_OBSERVE = 6
OT_COAP_OPTION_URI_PORT = 7
OT_COAP_OPTION_LOCATION_PATH = 8
OT_COAP_OPTION_URI_PATH = 11
OT_COAP_OPTION_CONTENT_FORMAT = 12
OT_COAP_OPTION_MAX_AGE = 14
OT_COAP_OPTION_URI_QUERY = 15
OT_COAP_OPTION_ACCEPT = 17
OT_COAP_OPTION_LOCATION_QUERY = 20
OT_COAP_OPTION_BLOCK2 = 23
OT_COAP_OPTION_BLOCK1 = 27
OT_COAP_OPTION_SIZE2 = 28
OT_COAP_OPTION_PROXY_URI = 35
OT_COAP_OPTION_PROXY_SCHEME = 39
OT_COAP_OPTION_SIZE1 = 60

}

CoAP Option Numbers.

enum otCoapOptionContentFormat {

OT_COAP_OPTION_CONTENT_FORMAT_TEXT_PLAIN = 0
OT_COAP_OPTION_CONTENT_FORMAT_COSE_ENCRYPT0 = 16
OT_COAP_OPTION_CONTENT_FORMAT_COSE_MAC0 = 17
OT_COAP_OPTION_CONTENT_FORMAT_COSE_SIGN1 = 18
OT_COAP_OPTION_CONTENT_FORMAT_LINK_FORMAT = 40
OT_COAP_OPTION_CONTENT_FORMAT_XML = 41
OT_COAP_OPTION_CONTENT_FORMAT_OCTET_STREAM = 42
OT_COAP_OPTION_CONTENT_FORMAT_EXI = 47
OT_COAP_OPTION_CONTENT_FORMAT_JSON = 50
OT_COAP_OPTION_CONTENT_FORMAT_JSON_PATCH_JSON = 51
OT_COAP_OPTION_CONTENT_FORMAT_MERGE_PATCH_JSON = 52
OT_COAP_OPTION_CONTENT_FORMAT_CBOR = 60
OT_COAP_OPTION_CONTENT_FORMAT_CWT = 61
OT_COAP_OPTION_CONTENT_FORMAT_COSE_ENCRYPT = 96
OT_COAP_OPTION_CONTENT_FORMAT_COSE_MAC = 97
OT_COAP_OPTION_CONTENT_FORMAT_COSE_SIGN = 98
OT_COAP_OPTION_CONTENT_FORMAT_COSE_KEY = 101
OT_COAP_OPTION_CONTENT_FORMAT_COSE_KEY_SET = 102
OT_COAP_OPTION_CONTENT_FORMAT_SENML_JSON = 110
OT_COAP_OPTION_CONTENT_FORMAT_SENSML_JSON = 111
OT_COAP_OPTION_CONTENT_FORMAT_SENML_CBOR = 112
OT_COAP_OPTION_CONTENT_FORMAT_SENSML_CBOR = 113
OT_COAP_OPTION_CONTENT_FORMAT_SENML_EXI = 114
OT_COAP_OPTION_CONTENT_FORMAT_SENSML_EXI = 115
OT_COAP_OPTION_CONTENT_FORMAT_COAP_GROUP_JSON = 256
OT_COAP_OPTION_CONTENT_FORMAT_SENML_XML = 310
OT_COAP_OPTION_CONTENT_FORMAT_SENSML_XML = 311

}

CoAP Content Format codes.

enum otCoapBlockSzx {

OT_COAP_OPTION_BLOCK_SZX_16 = 0
OT_COAP_OPTION_BLOCK_SZX_32 = 1
OT_COAP_OPTION_BLOCK_SZX_64 = 2
OT_COAP_OPTION_BLOCK_SZX_128 = 3
OT_COAP_OPTION_BLOCK_SZX_256 = 4
OT_COAP_OPTION_BLOCK_SZX_512 = 5
OT_COAP_OPTION_BLOCK_SZX_1024 = 6

}

CoAP Block Size Exponents.

Typedefs

typedef enum otCoapType — otCoapType
CoAP Type values (2 bit unsigned integer).

typedef enum otCoapCode — otCoapCode
CoAP Code values.

typedef enum otCoapOptionType — otCoapOptionType
CoAP Option Numbers.

typedef struct otCoapOption — otCoapOption
Represents a CoAP option.

typedef struct otCoapOptionIterator — otCoapOptionIterator
Acts as an iterator for CoAP options.

typedef enum otCoapOptionContentFormat
CoAP Content Format codes.

typedef enum otCoapBlockSzx
CoAP Block Size Exponents.

typedef void(* otCoapResponseHandler)(void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo, otError aResult)
Pointer is called when a CoAP response is received or on the request timeout.

typedef void(* otCoapRequestHandler)(void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo)
Pointer is called when a CoAP request with a given Uri-Path is received.

typedef otError(* otCoapBlockwiseReceiveHook)(void *aContext, const uint8_t *aBlock, uint32_t aPosition, uint16_t aBlockLength, bool aMore, uint32_t aTotalLength)
Pointer is called when a CoAP message with an block-wise transfer option is received.

typedef otError(* otCoapBlockwiseTransmitHook)(void *aContext, uint8_t *aBlock, uint32_t aPosition, uint16_t *aBlockLength, bool *aMore)
Pointer is called before the next block in a block-wise transfer is sent.

typedef struct otCoapResource
Represents a CoAP resource.

typedef struct otCoapBlockwiseResource
Represents a CoAP resource with block-wise transfer.

typedef struct otCoapTxParameters
Represents the CoAP transmission parameters.

Functions

void otCoapMessageInit(otMessage *aMessage, otCoapType aType, otCoapCode aCode)
Initializes the CoAP header.

otError otCoapMessageInitResponse(otMessage *aResponse, const otMessage *aRequest, otCoapType aType, otCoapCode aCode)
Initializes a response message.

otError otCoapMessageSetToken(otMessage *aMessage, const uint8_t *aToken, uint8_t aTokenLength)
Sets the Token value and length in a header.

void otCoapMessageGenerateToken(otMessage *aMessage, uint8_t aTokenLength)
Sets the Token length and randomizes its value.

otError otCoapMessageAppendContentFormatOption(otMessage *aMessage, otCoapOptionContentFormat aContentFormat)
Appends the Content Format CoAP option as specified in https://tools.ietf.org/html/rfc7252#page-92.

otError otCoapMessageAppendOption(otMessage *aMessage, uint16_t aNumber, uint16_t aLength, const void *aValue)
Appends a CoAP option in a header.

otError otCoapMessageAppendUintOption(otMessage *aMessage, uint16_t aNumber, uint32_t aValue)
Appends an unsigned integer CoAP option as specified in https://tools.ietf.org/html/rfc7252#section-3.2.

otError otCoapMessageAppendObserveOption(otMessage *aMessage, uint32_t aObserve)
Appends an Observe option.

otError	otCoapMessageAppendUriPathOptions(otMessage *aMessage, const char *aUriPath) Appends a Uri-Path option.
uint16_t	otCoapBlockSizeFromExponent(otCoapBlockSzx aSize) Converts a CoAP Block option SZX field to the actual block size.
otError	otCoapMessageAppendBlock2Option(otMessage *aMessage, uint32_t aNum, bool aMore, otCoapBlockSzx aSize) Appends a Block2 option.
otError	otCoapMessageAppendBlock1Option(otMessage *aMessage, uint32_t aNum, bool aMore, otCoapBlockSzx aSize) Appends a Block1 option.
otError	otCoapMessageAppendProxyUriOption(otMessage *aMessage, const char *aUriPath) Appends a Proxy-Uri option.
otError	otCoapMessageAppendMaxAgeOption(otMessage *aMessage, uint32_t aMaxAge) Appends a Max-Age option.
otError	otCoapMessageAppendUriQueryOption(otMessage *aMessage, const char *aUriQuery) Appends a single Uri-Query option.
otError	otCoapMessageSetPayloadMarker(otMessage *aMessage) Adds Payload Marker indicating beginning of the payload to the CoAP header.
otCoapType	otCoapMessageGetType(const otMessage *aMessage) Returns the Type value.
otCoapCode	otCoapMessageGetCode(const otMessage *aMessage) Returns the Code value.
void	otCoapMessageSetCode(otMessage *aMessage, otCoapCode aCode) Sets the Code value.
const char *	otCoapMessageCodeToString(const otMessage *aMessage) Returns the CoAP Code as human readable string.
uint16_t	otCoapMessageGetMessageId(const otMessage *aMessage) Returns the Message ID value.
uint8_t	otCoapMessageGetTokenLength(const otMessage *aMessage) Returns the Token length.
const uint8_t *	otCoapMessageGetToken(const otMessage *aMessage) Returns a pointer to the Token value.
otError	otCoapOptionIteratorInit(otCoapOptionIterator *aIterator, const otMessage *aMessage) Initialises an iterator for the options in the given message.
const otCoapOption *	otCoapOptionIteratorGetFirstOptionMatching(otCoapOptionIterator *aIterator, uint16_t aOption) Returns a pointer to the first option matching the specified option number.
const otCoapOption *	otCoapOptionIteratorGetFirstOption(otCoapOptionIterator *aIterator) Returns a pointer to the first option.
const otCoapOption *	otCoapOptionIteratorGetNextOptionMatching(otCoapOptionIterator *aIterator, uint16_t aOption) Returns a pointer to the next option matching the specified option number.
const otCoapOption *	otCoapOptionIteratorGetNextOption(otCoapOptionIterator *aIterator) Returns a pointer to the next option.

otError otCoapOptionIteratorGetOptionUintValue(otCoapOptionIterator *aIterator, uint64_t *aValue)
Fills current option value into `aValue` assuming the current value is an unsigned integer encoded according to https://tools.ietf.org/html/rfc7252#section-3.2.

otError otCoapOptionIteratorGetOptionValue(otCoapOptionIterator *aIterator, void *aValue)
Fills current option value into `aValue`.

otMessage * otCoapNewMessage(otInstance *aInstance, const otMessageSettings *aSettings)
Creates a new CoAP message.

otError otCoapSendRequestWithParameters(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, otCoapResponseHandler aHandler, void *aContext, const otCoapTxParameters *aTxParameters)
Sends a CoAP request with custom transmission parameters.

otError otCoapSendRequestBlockWiseWithParameters(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, otCoapResponseHandler aHandler, void *aContext, const otCoapTxParameters *aTxParameters, otCoapBlockwiseTransmitHook aTransmitHook, otCoapBlockwiseReceiveHook aReceiveHook)
Sends a CoAP request block-wise with custom transmission parameters.

otError otCoapSendRequestBlockWise(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, otCoapResponseHandler aHandler, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook, otCoapBlockwiseReceiveHook aReceiveHook)
Sends a CoAP request block-wise.

otError otCoapSendRequest(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, otCoapResponseHandler aHandler, void *aContext)
Sends a CoAP request.

otError otCoapStart(otInstance *aInstance, uint16_t aPort)
Starts the CoAP server.

otError otCoapStop(otInstance *aInstance)
Stops the CoAP server.

void otCoapAddResource(otInstance *aInstance, otCoapResource *aResource)
Adds a resource to the CoAP server.

void otCoapRemoveResource(otInstance *aInstance, otCoapResource *aResource)
Removes a resource from the CoAP server.

void otCoapAddBlockWiseResource(otInstance *aInstance, otCoapBlockwiseResource *aResource)
Adds a block-wise resource to the CoAP server.

void otCoapRemoveBlockWiseResource(otInstance *aInstance, otCoapBlockwiseResource *aResource)
Removes a block-wise resource from the CoAP server.

void otCoapSetDefaultHandler(otInstance *aInstance, otCoapRequestHandler aHandler, void *aContext)
Sets the default handler for unhandled CoAP requests.

otError otCoapSendResponseWithParameters(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, const otCoapTxParameters *aTxParameters)
Sends a CoAP response from the server with custom transmission parameters.

otError otCoapSendResponseBlockWiseWithParameters(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, const otCoapTxParameters *aTxParameters, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook)
Sends a CoAP response block-wise from the server with custom transmission parameters.

otError otCoapSendResponseBlockWise(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook)
Sends a CoAP response block-wise from the server.

otError otCoapSendResponse(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo)
Sends a CoAP response from the server.

Macros

#define OT_DEFAULT_COAP_PORT 5683
Default CoAP port, as specified in RFC 7252.

#define OT_COAP_DEFAULT_TOKEN_LENGTH 2
Default token length.

#define OT_COAP_MAX_TOKEN_LENGTH 8
Max token length as specified (RFC 7252).

#define OT_COAP_MAX_RETRANSMIT 20
Max retransmit supported by OpenThread.

#define OT_COAP_MIN_ACK_TIMEOUT 1000
Minimal ACK timeout in milliseconds supported by OpenThread.

#define OT_COAP_CODE (c, d)
Helper macro to define CoAP Code values.

Enumeration Documentation

otCoapType

> otCoapType

CoAP Type values (2 bit unsigned integer).

Enumerator	
OT_COAP_TYPE_CONFIRMABLE	Confirmable.
OT_COAP_TYPE_NON_CONFIRMABLE	Non-confirmable.
OT_COAP_TYPE_ACKNOWLEDGMENT	Acknowledgment.
OT_COAP_TYPE_RESET	Reset.

Definition at line 73 of file include/openthread/coap.h

otCoapCode

> otCoapCode

CoAP Code values.

Enumerator	
OT_COAP_CODE_EMPTY	Empty message code.
OT_COAP_CODE_GET	Get.
OT_COAP_CODE_POST	Post.
OT_COAP_CODE_PUT	Put.
OT_COAP_CODE_DELETE	Delete.
OT_COAP_CODE_RESPONSE_MIN	2.00
OT_COAP_CODE_CREATED	Created.

OT_COAP_CODE_DELETED	Deleted.
OT_COAP_CODE_VALID	Valid.
OT_COAP_CODE_CHANGED	Changed.
OT_COAP_CODE_CONTENT	Content.
OT_COAP_CODE_CONTINUE	RFC7959 Continue.
OT_COAP_CODE_BAD_REQUEST	Bad Request.
OT_COAP_CODE_UNAUTHORIZED	Unauthorized.
OT_COAP_CODE_BAD_OPTION	Bad Option.
OT_COAP_CODE_FORBIDDEN	Forbidden.
OT_COAP_CODE_NOT_FOUND	Not Found.
OT_COAP_CODE_METHOD_NOT_ALLOWED	Method Not Allowed.
OT_COAP_CODE_NOT_ACCEPTABLE	Not Acceptable.
OT_COAP_CODE_REQUEST_INCOMPLETE	RFC7959 Request Entity Incomplete.
OT_COAP_CODE_PRECONDITION_FAILED	Precondition Failed.
OT_COAP_CODE_REQUEST_TOO_LARGE	Request Entity Too Large.
OT_COAP_CODE_UNSUPPORTED_FORMAT	Unsupported Content-Format.
OT_COAP_CODE_INTERNAL_ERROR	Internal Server Error.
OT_COAP_CODE_NOT_IMPLEMENTED	Not Implemented.
OT_COAP_CODE_BAD_GATEWAY	Bad Gateway.
OT_COAP_CODE_SERVICE_UNAVAILABLE	Service Unavailable.
OT_COAP_CODE_GATEWAY_TIMEOUT	Gateway Timeout.
OT_COAP_CODE_PROXY_NOT_SUPPORTED	Proxying Not Supported.

Definition at line `91` of file `include/openthread/coap.h`

otCoapOptionType

> otCoapOptionType

CoAP Option Numbers.

Enumerator	
OT_COAP_OPTION_IF_MATCH	If-Match.
OT_COAP_OPTION_URI_HOST	Uri-Host.
OT_COAP_OPTION_E_TAG	ETag.
OT_COAP_OPTION_IF_NONE_MATCH	If-None-Match.
OT_COAP_OPTION_OBSERVE	Observe [RFC7641].
OT_COAP_OPTION_URI_PORT	Uri-Port.
OT_COAP_OPTION_LOCATION_PATH	Location-Path.
OT_COAP_OPTION_URI_PATH	Uri-Path.
OT_COAP_OPTION_CONTENT_FORMAT	Content-Format.
OT_COAP_OPTION_MAX_AGE	Max-Age.
OT_COAP_OPTION_URI_QUERY	Uri-Query.
OT_COAP_OPTION_ACCEPT	Accept.

OT_COAP_OPTION_LOCATION_QUERY	Location-Query.
OT_COAP_OPTION_BLOCK2	Block2 (RFC7959)
OT_COAP_OPTION_BLOCK1	Block1 (RFC7959)
OT_COAP_OPTION_SIZE2	Size2 (RFC7959)
OT_COAP_OPTION_PROXY_URI	Proxy-Uri.
OT_COAP_OPTION_PROXY_SCHEME	Proxy-Scheme.
OT_COAP_OPTION_SIZE1	Size1.

Definition at line 130 of file include/openthread/coap.h

otCoapOptionContentFormat

> otCoapOptionContentFormat

CoAP Content Format codes.

The full list is documented at https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats

Enumerator	
OT_COAP_OPTION_CONTENT_FORMAT_TEXT_PLAIN	text/plain; charset=utf-8: [RFC2046][RFC3676][RFC5147]
OT_COAP_OPTION_CONTENT_FORMAT_COSE_ENCRYPT0	application/cose; cose-type="cose-encrypt0": [RFC8152]
OT_COAP_OPTION_CONTENT_FORMAT_COSE_MAC0	application/cose; cose-type="cose-mac0": [RFC8152]
OT_COAP_OPTION_CONTENT_FORMAT_COSE_SIGN1	application/cose; cose-type="cose-sign1": [RFC8152]
OT_COAP_OPTION_CONTENT_FORMAT_LINK_FORMAT	application/link-format: [RFC6690]
OT_COAP_OPTION_CONTENT_FORMAT_XML	application/xml: [RFC3023]
OT_COAP_OPTION_CONTENT_FORMAT_OCTET_STREAM	application/octet-stream: [RFC2045][RFC2046]
OT_COAP_OPTION_CONTENT_FORMAT_EXI	application/exi: ["Efficient XML Interchange (EXI) Format 1.0 (Second Edition)", February 2014]
OT_COAP_OPTION_CONTENT_FORMAT_JSON	application/json: [RFC7159]
OT_COAP_OPTION_CONTENT_FORMAT_JSON_PATCH_JSON	application/json-patch+json: [RFC6902]
OT_COAP_OPTION_CONTENT_FORMAT_MERGE_PATCH_JSON	application/merge-patch+json: [RFC7396]
OT_COAP_OPTION_CONTENT_FORMAT_CBOR	application/cbor: [RFC7049]
OT_COAP_OPTION_CONTENT_FORMAT_CWT	application/cwt: [RFC8392]
OT_COAP_OPTION_CONTENT_FORMAT_COSE_ENCRYPT	application/cose; cose-type="cose-encrypt": [RFC8152]
OT_COAP_OPTION_CONTENT_FORMAT_COSE_MAC	application/cose; cose-type="cose-mac": [RFC8152]
OT_COAP_OPTION_CONTENT_FORMAT_COSE_SIGN	application/cose; cose-type="cose-sign": [RFC8152]
OT_COAP_OPTION_CONTENT_FORMAT_COSE_KEY	application/cose-key: [RFC8152]
OT_COAP_OPTION_CONTENT_FORMAT_COSE_KEY_SET	application/cose-key-set: [RFC8152]
OT_COAP_OPTION_CONTENT_FORMAT_SENML_JSON	application/senml+json: [RFC8428]
OT_COAP_OPTION_CONTENT_FORMAT_SENSML_JSON	application/sensml+json: [RFC8428]
OT_COAP_OPTION_CONTENT_FORMAT_SENML_CBOR	application/senml+cbor: [RFC8428]
OT_COAP_OPTION_CONTENT_FORMAT_SENSML_CBOR	application/sensml+cbor: [RFC8428]
OT_COAP_OPTION_CONTENT_FORMAT_SENML_EXI	application/senml-exi: [RFC8428]
OT_COAP_OPTION_CONTENT_FORMAT_SENSML_EXI	application/sensml-exi: [RFC8428]
OT_COAP_OPTION_CONTENT_FORMAT_COAP_GROUP_JSON	application/coap-group+json: [RFC7390]

OT_COAP_OPTION_CONTENT_FORMAT_SENML_XML	application/senml+xml: [RFC8428]
OT_COAP_OPTION_CONTENT_FORMAT_SENSML_XML	application/sensml+xml: [RFC8428]

Definition at line 178 of file include/openthread/coap.h

otCoapBlockSzx

```
otCoapBlockSzx
```

CoAP Block Size Exponents.

Enumerator	
OT_COAP_OPTION_BLOCK_SZX_16	
OT_COAP_OPTION_BLOCK_SZX_32	
OT_COAP_OPTION_BLOCK_SZX_64	
OT_COAP_OPTION_BLOCK_SZX_128	
OT_COAP_OPTION_BLOCK_SZX_256	
OT_COAP_OPTION_BLOCK_SZX_512	
OT_COAP_OPTION_BLOCK_SZX_1024	

Definition at line 320 of file include/openthread/coap.h

Typedef Documentation

otCoapType

```
typedef enum otCoapType otCoapType
```

CoAP Type values (2 bit unsigned integer).

Definition at line 79 of file include/openthread/coap.h

otCoapCode

```
typedef enum otCoapCode otCoapCode
```

CoAP Code values.

Definition at line 125 of file include/openthread/coap.h

otCoapOptionType

```
typedef enum otCoapOptionType otCoapOptionType
```

CoAP Option Numbers.

Definition at line 151 of file include/openthread/coap.h

otCoapOption

```
typedef struct otCoapOption otCoapOption
```

Represents a CoAP option.

Definition at line `161` of file `include/openthread/coap.h`

otCoapOptionIterator

```
typedef struct otCoapOptionIterator otCoapOptionIterator
```

Acts as an iterator for CoAP options.

Definition at line `172` of file `include/openthread/coap.h`

otCoapOptionContentFormat

```
typedef enum otCoapOptionContentFormat otCoapOptionContentFormat
```

CoAP Content Format codes.

The full list is documented at https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats

Definition at line `315` of file `include/openthread/coap.h`

otCoapBlockSzx

```
typedef enum otCoapBlockSzx otCoapBlockSzx
```

CoAP Block Size Exponents.

Definition at line `329` of file `include/openthread/coap.h`

otCoapResponseHandler

```
typedef void(* otCoapResponseHandler) (void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo,
otError aResult) )(void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo, otError aResult)
```

Pointer is called when a CoAP response is received or on the request timeout.

Parameters

[in]	aContext	A pointer to application-specific context.
[in]	aMessage	A pointer to the message buffer containing the response. NULL if no response was received.
[in]	aMessageInfo	A pointer to the message info for `aMessage` . NULL if no response was received.
[in]	aResult	A result of the CoAP transaction.

Definition at line `344` of file `include/openthread/coap.h`

otCoapRequestHandler

```
typedef void(* otCoapRequestHandler) (void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo) )
(void *aContext, otMessage *aMessage, const otMessageInfo *aMessageInfo)
```

Pointer is called when a CoAP request with a given Uri-Path is received.

Parameters

[in]	aContext	A pointer to arbitrary context information.
[in]	aMessage	A pointer to the message.
[in]	aMessageInfo	A pointer to the message info for `aMessage`.

Definition at line `357` of file `include/openthread/coap.h`

otCoapBlockwiseReceiveHook

```
typedef otError(* otCoapBlockwiseReceiveHook) (void *aContext, const uint8_t *aBlock, uint32_t aPosition, uint16_t
aBlockLength, bool aMore, uint32_t aTotalLength) )(void *aContext, const uint8_t *aBlock, uint32_t aPosition, uint16_t
aBlockLength, bool aMore, uint32_t aTotalLength)
```

Pointer is called when a CoAP message with an block-wise transfer option is received.

Parameters

[in]	aContext	A pointer to application-specific context.
[in]	aBlock	A pointer to the block segment.
[in]	aPosition	The position of `aBlock` in a sequence in bytes.
[in]	aBlockLength	The length of the block segment in bytes.
[in]	aMore	Flag if more block segments are following.
[in]	aTotalLength	The total length in bytes of the transferred information (indicated by a Size1 or Size2 option).

Is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

Definition at line `378` of file `include/openthread/coap.h`

otCoapBlockwiseTransmitHook

```
typedef otError(* otCoapBlockwiseTransmitHook) (void *aContext, uint8_t *aBlock, uint32_t aPosition, uint16_t
*aBlockLength, bool *aMore) )(void *aContext, uint8_t *aBlock, uint32_t aPosition, uint16_t *aBlockLength, bool *aMore)
```

Pointer is called before the next block in a block-wise transfer is sent.

Parameters

[in]	aContext	A pointer to application-specific context.
[inout]	aBlock	A pointer to where the block segment can be written to.
[in]	aPosition	The position in a sequence from which to obtain the block segment.
[inout]	aBlockLength	On entry, the maximum block segment length in bytes.
[out]	aMore	A pointer to the flag if more block segments will follow.

Is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

Warnings

By changing the value of aBlockLength, the block size of the whole exchange is renegotiated. It is recommended to do this after the first block has been received as later changes could cause problems with other CoAP implementations.

Definition at line 405 of file include/openthread/coap.h

otCoapResource

```
typedef struct otCoapResource otCoapResource
```

Represents a CoAP resource.

Definition at line 421 of file include/openthread/coap.h

otCoapBlockwiseResource

```
typedef struct otCoapBlockwiseResource otCoapBlockwiseResource
```

Represents a CoAP resource with block-wise transfer.

Definition at line 445 of file include/openthread/coap.h

otCoapTxParameters

```
typedef struct otCoapTxParameters otCoapTxParameters
```

Represents the CoAP transmission parameters.

Note

- mAckTimeout * ((2 ** (mMaxRetransmit + 1)) - 1) * (mAckRandomFactorNumerator / mAckRandomFactorDenominator) must not exceed what can be represented by a uint32_t (0xffffffff). This limitation allows OpenThread to avoid 64-bit arithmetic.

Definition at line 483 of file include/openthread/coap.h

Function Documentation

otCoapMessageInit

```
void otCoapMessageInit (otMessage *aMessage, otCoapType aType, otCoapCode aCode)
```

Initializes the CoAP header.

Parameters

[inout]	aMessage	A pointer to the CoAP message to initialize.
[in]	aType	CoAP message type.
[in]	aCode	CoAP message code.

Definition at line 493 of file include/openthread/coap.h

otCoapMessageInitResponse

> otError otCoapMessageInitResponse (otMessage *aResponse, const otMessage *aRequest, otCoapType aType, otCoapCode aCode)

Initializes a response message.

Parameters

[inout]	aResponse	A pointer to the CoAP response message.
[in]	aRequest	A pointer to the CoAP request message.
[in]	aType	CoAP message type.
[in]	aCode	CoAP message code.

Note

- Both message ID and token are set according to `aRequest`.

Definition at line `509` of file `include/openthread/coap.h`

otCoapMessageSetToken

> otError otCoapMessageSetToken (otMessage *aMessage, const uint8_t *aToken, uint8_t aTokenLength)

Sets the Token value and length in a header.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aToken	A pointer to the Token value.
[in]	aTokenLength	The Length of `aToken`.

Definition at line `522` of file `include/openthread/coap.h`

otCoapMessageGenerateToken

> void otCoapMessageGenerateToken (otMessage *aMessage, uint8_t aTokenLength)

Sets the Token length and randomizes its value.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aTokenLength	The Length of a Token to set.

Definition at line `531` of file `include/openthread/coap.h`

otCoapMessageAppendContentFormatOption

> otError otCoapMessageAppendContentFormatOption (otMessage *aMessage, otCoapOptionContentFormat aContentFormat)

Appends the Content Format CoAP option as specified in https://tools.ietf.org/html/rfc7252#page-92.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aContentFormat	One of the content formats listed in otCoapOptionContentFormat above.

This **must** be called before setting otCoapMessageSetPayloadMarker if a payload is to be included in the message.

The function is a convenience wrapper around otCoapMessageAppendUintOption, and if the desired format type code isn't listed in otCoapOptionContentFormat, this base function should be used instead.

Definition at line `552` of file `include/openthread/coap.h`

otCoapMessageAppendOption

> otError otCoapMessageAppendOption (otMessage *aMessage, uint16_t aNumber, uint16_t aLength, const void *aValue)

Appends a CoAP option in a header.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aNumber	The CoAP Option number.
[in]	aLength	The CoAP Option length.
[in]	aValue	A pointer to the CoAP value.

Definition at line `567` of file `include/openthread/coap.h`

otCoapMessageAppendUintOption

> otError otCoapMessageAppendUintOption (otMessage *aMessage, uint16_t aNumber, uint32_t aValue)

Appends an unsigned integer CoAP option as specified in https://tools.ietf.org/html/rfc7252#section-3.2.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aNumber	The CoAP Option number.
[in]	aValue	The CoAP Option unsigned integer value.

See Also

- otCoapMessageGetOptionUintValue

Definition at line `583` of file `include/openthread/coap.h`

otCoapMessageAppendObserveOption

> otError otCoapMessageAppendObserveOption (otMessage *aMessage, uint32_t aObserve)

Appends an Observe option.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aObserve	Observe field value.

Definition at line 596 of file include/openthread/coap.h

otCoapMessageAppendUriPathOptions

> otError otCoapMessageAppendUriPathOptions (otMessage *aMessage, const char *aUriPath)

Appends a Uri-Path option.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aUriPath	A pointer to a NULL-terminated string.

Definition at line 609 of file include/openthread/coap.h

otCoapBlockSizeFromExponent

> uint16_t otCoapBlockSizeFromExponent (otCoapBlockSzx aSize)

Converts a CoAP Block option SZX field to the actual block size.

Parameters

[in]	aSize	Block size exponent.

Returns

- The actual size exponent value.

Definition at line 619 of file include/openthread/coap.h

otCoapMessageAppendBlock2Option

> otError otCoapMessageAppendBlock2Option (otMessage *aMessage, uint32_t aNum, bool aMore, otCoapBlockSzx aSize)

Appends a Block2 option.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aNum	Current block number.
[in]	aMore	Boolean to indicate more blocks are to be sent.
[in]	aSize	Block Size Exponent.

Definition at line 634 of file include/openthread/coap.h

otCoapMessageAppendBlock1Option

> otError otCoapMessageAppendBlock1Option (otMessage *aMessage, uint32_t aNum, bool aMore, otCoapBlockSzx aSize)

Appends a Block1 option.

Parameters

[inout]	aMessage	A pointer to the CoAP message.

[in]	aNum	Current block number.
[in]	aMore	Boolean to indicate more blocks are to be sent.
[in]	aSize	Block Size Exponent.

Definition at line `649` of file `include/openthread/coap.h`

otCoapMessageAppendProxyUriOption

> otError otCoapMessageAppendProxyUriOption (otMessage *aMessage, const char *aUriPath)

Appends a Proxy-Uri option.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aUriPath	A pointer to a NULL-terminated string.

Definition at line `662` of file `include/openthread/coap.h`

otCoapMessageAppendMaxAgeOption

> otError otCoapMessageAppendMaxAgeOption (otMessage *aMessage, uint32_t aMaxAge)

Appends a Max-Age option.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aMaxAge	The Max-Age value.

Definition at line `675` of file `include/openthread/coap.h`

otCoapMessageAppendUriQueryOption

> otError otCoapMessageAppendUriQueryOption (otMessage *aMessage, const char *aUriQuery)

Appends a single Uri-Query option.

Parameters

[inout]	aMessage	A pointer to the CoAP message.
[in]	aUriQuery	A pointer to NULL-terminated string, which should contain a single key=value pair.

Definition at line `687` of file `include/openthread/coap.h`

otCoapMessageSetPayloadMarker

> otError otCoapMessageSetPayloadMarker (otMessage *aMessage)

Adds Payload Marker indicating beginning of the payload to the CoAP header.

Parameters

[inout]	aMessage	A pointer to the CoAP message.

Definition at line 698 of file include/openthread/coap.h

otCoapMessageGetType

```
otCoapType otCoapMessageGetType (const otMessage *aMessage)
```

Returns the Type value.

Parameters

[in]	aMessage	A pointer to the CoAP message.

Returns

- The Type value.

Definition at line 708 of file include/openthread/coap.h

otCoapMessageGetCode

```
otCoapCode otCoapMessageGetCode (const otMessage *aMessage)
```

Returns the Code value.

Parameters

[in]	aMessage	A pointer to the CoAP message.

Returns

- The Code value.

Definition at line 718 of file include/openthread/coap.h

otCoapMessageSetCode

```
void otCoapMessageSetCode (otMessage *aMessage, otCoapCode aCode)
```

Sets the Code value.

Parameters

[inout]	aMessage	A pointer to the CoAP message to initialize.
[in]	aCode	CoAP message code.

Definition at line 727 of file include/openthread/coap.h

otCoapMessageCodeToString

```
const char * otCoapMessageCodeToString (const otMessage *aMessage)
```

Returns the CoAP Code as human readable string.

Parameters

[in]	aMessage	A pointer to the CoAP message.

@ returns The CoAP Code as string.

Definition at line 737 of file include/openthread/coap.h

otCoapMessageGetMessageId

```
uint16_t otCoapMessageGetMessageId (const otMessage *aMessage)
```

Returns the Message ID value.

Parameters

[in]	aMessage	A pointer to the CoAP message.

Returns

- The Message ID value.

Definition at line 747 of file include/openthread/coap.h

otCoapMessageGetTokenLength

```
uint8_t otCoapMessageGetTokenLength (const otMessage *aMessage)
```

Returns the Token length.

Parameters

[in]	aMessage	A pointer to the CoAP message.

Returns

- The Token length.

Definition at line 757 of file include/openthread/coap.h

otCoapMessageGetToken

```
const uint8_t * otCoapMessageGetToken (const otMessage *aMessage)
```

Returns a pointer to the Token value.

Parameters

[in]	aMessage	A pointer to the CoAP message.

Returns

- A pointer to the Token value.

Definition at line 767 of file include/openthread/coap.h

otCoapOptionIteratorInit

> otError otCoapOptionIteratorInit (otCoapOptionIterator *aIterator, const otMessage *aMessage)

Initialises an iterator for the options in the given message.

Parameters

[inout]	aIterator	A pointer to the CoAP message option iterator.
[in]	aMessage	A pointer to the CoAP message.

Definition at line `779` of file `include/openthread/coap.h`

otCoapOptionIteratorGetFirstOptionMatching

> const otCoapOption * otCoapOptionIteratorGetFirstOptionMatching (otCoapOptionIterator *aIterator, uint16_t aOption)

Returns a pointer to the first option matching the specified option number.

Parameters

[in]	aIterator	A pointer to the CoAP message option iterator.
[in]	aOption	The option number sought.

Returns

- A pointer to the first matching option. If no matching option is present NULL pointer is returned.

Definition at line `790` of file `include/openthread/coap.h`

otCoapOptionIteratorGetFirstOption

> const otCoapOption * otCoapOptionIteratorGetFirstOption (otCoapOptionIterator *aIterator)

Returns a pointer to the first option.

Parameters

[inout]	aIterator	A pointer to the CoAP message option iterator.

Returns

- A pointer to the first option. If no option is present NULL pointer is returned.

Definition at line `800` of file `include/openthread/coap.h`

otCoapOptionIteratorGetNextOptionMatching

> const otCoapOption * otCoapOptionIteratorGetNextOptionMatching (otCoapOptionIterator *aIterator, uint16_t aOption)

Returns a pointer to the next option matching the specified option number.

Parameters

[in]	aIterator	A pointer to the CoAP message option iterator.
[in]	aOption	The option number sought.

Returns

- A pointer to the next matching option. If no further matching option is present NULL pointer is returned.

Definition at line 811 of file include/openthread/coap.h

otCoapOptionIteratorGetNextOption

```
const otCoapOption * otCoapOptionIteratorGetNextOption (otCoapOptionIterator *aIterator)
```

Returns a pointer to the next option.

Parameters

[inout]	aIterator	A pointer to the CoAP message option iterator.

Returns

- A pointer to the next option. If no more options are present NULL pointer is returned.

Definition at line 821 of file include/openthread/coap.h

otCoapOptionIteratorGetOptionUintValue

```
otError otCoapOptionIteratorGetOptionUintValue (otCoapOptionIterator *aIterator, uint64_t *aValue)
```

Fills current option value into aValue assuming the current value is an unsigned integer encoded according to https://tools.ietf.org/html/rfc7252#section-3.2.

Parameters

[inout]	aIterator	A pointer to the CoAP message option iterator.
[out]	aValue	A pointer to an unsigned integer to receive the option value.

See Also

- otCoapMessageAppendUintOption

Definition at line 836 of file include/openthread/coap.h

otCoapOptionIteratorGetOptionValue

```
otError otCoapOptionIteratorGetOptionValue (otCoapOptionIterator *aIterator, void *aValue)
```

Fills current option value into aValue.

Parameters

[inout]	aIterator	A pointer to the CoAP message option iterator.
[out]	aValue	A pointer to a buffer to receive the option value.

Definition at line 848 of file include/openthread/coap.h

otCoapNewMessage

> otMessage * otCoapNewMessage (otInstance *aInstance, const otMessageSettings *aSettings)

Creates a new CoAP message.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSettings	A pointer to the message settings or NULL to set default settings.

Note

- If `aSettings` is 'NULL', the link layer security is enabled and the message priority is set to OT_MESSAGE_PRIORITY_NORMAL by default.

Returns

- A pointer to the message buffer or NULL if no message buffers are available or parameters are invalid.

Definition at line `862` of file `include/openthread/coap.h`

otCoapSendRequestWithParameters

> otError otCoapSendRequestWithParameters (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, otCoapResponseHandler aHandler, void *aContext, const otCoapTxParameters *aTxParameters)

Sends a CoAP request with custom transmission parameters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the message to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.
[in]	aHandler	A function pointer that shall be called on response reception or timeout.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.
[in]	aTxParameters	A pointer to transmission parameters for this request. Use NULL for defaults. Otherwise, parameters given must meet the following conditions: 1. mMaxRetransmit is no more than OT_COAP_MAX_RETRANSMIT. 2. mAckRandomFactorNumerator / mAckRandomFactorDenominator must not be below 1.0. 3. The calculated exchange life time must not overflow uint32_t.

If a response for a request is expected, respective function and context information should be provided. If no response is expected, these arguments should be NULL pointers.

Definition at line `886` of file `include/openthread/coap.h`

otCoapSendRequestBlockWiseWithParameters

> otError otCoapSendRequestBlockWiseWithParameters (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, otCoapResponseHandler aHandler, void *aContext, const otCoapTxParameters *aTxParameters, otCoapBlockwiseTransmitHook aTransmitHook, otCoapBlockwiseReceiveHook aReceiveHook)

Sends a CoAP request block-wise with custom transmission parameters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the message to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.
[in]	aHandler	A function pointer that shall be called on response reception or timeout.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.
[in]	aTxParameters	A pointer to transmission parameters for this request. Use NULL for defaults.
[in]	aTransmitHook	A pointer to a hook function for outgoing block-wise transfer.
[in]	aReceiveHook	A pointer to a hook function for incoming block-wise transfer.

Is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

If a response for a request is expected, respective function and context information should be provided. If the response is expected to be block-wise, a respective hook function should be provided. If no response is expected, these arguments should be NULL pointers.

Definition at line `917` of file `include/openthread/coap.h`

otCoapSendRequestBlockWise

> static otError otCoapSendRequestBlockWise (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, otCoapResponseHandler aHandler, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook, otCoapBlockwiseReceiveHook aReceiveHook)

Sends a CoAP request block-wise.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the message to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.
[in]	aHandler	A function pointer that shall be called on response reception or timeout.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.
[in]	aTransmitHook	A pointer to a hook function for outgoing block-wise transfer.
[in]	aReceiveHook	A pointer to a hook function for incoming block-wise transfer.

Is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

If a response for a request is expected, respective function and context information should be provided. If the response is expected to be block-wise, a respective hook function should be provided. If no response is expected, these arguments should be NULL pointers.

Definition at line `948` of file `include/openthread/coap.h`

otCoapSendRequest

> static otError otCoapSendRequest (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, otCoapResponseHandler aHandler, void *aContext)

Sends a CoAP request.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[in]	aMessage	A pointer to the message to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.
[in]	aHandler	A function pointer that shall be called on response reception or timeout.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.

If a response for a request is expected, respective function and context information should be provided. If no response is expected, these arguments should be NULL pointers.

Definition at line `977` of file `include/openthread/coap.h`

otCoapStart

```
otError otCoapStart (otInstance *aInstance, uint16_t aPort)
```

Starts the CoAP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPort	The local UDP port to bind to.

Definition at line `997` of file `include/openthread/coap.h`

otCoapStop

```
otError otCoapStop (otInstance *aInstance)
```

Stops the CoAP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `1007` of file `include/openthread/coap.h`

otCoapAddResource

```
void otCoapAddResource (otInstance *aInstance, otCoapResource *aResource)
```

Adds a resource to the CoAP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aResource	A pointer to the resource.

Definition at line `1016` of file `include/openthread/coap.h`

otCoapRemoveResource

```
void otCoapRemoveResource (otInstance *aInstance, otCoapResource *aResource)
```

Removes a resource from the CoAP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aResource	A pointer to the resource.

Definition at line `1025` of file `include/openthread/coap.h`

otCoapAddBlockWiseResource

> void otCoapAddBlockWiseResource (otInstance *aInstance, otCoapBlockwiseResource *aResource)

Adds a block-wise resource to the CoAP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aResource	A pointer to the resource.

Definition at line `1034` of file `include/openthread/coap.h`

otCoapRemoveBlockWiseResource

> void otCoapRemoveBlockWiseResource (otInstance *aInstance, otCoapBlockwiseResource *aResource)

Removes a block-wise resource from the CoAP server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aResource	A pointer to the resource.

Definition at line `1043` of file `include/openthread/coap.h`

otCoapSetDefaultHandler

> void otCoapSetDefaultHandler (otInstance *aInstance, otCoapRequestHandler aHandler, void *aContext)

Sets the default handler for unhandled CoAP requests.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aHandler	A function pointer that shall be called when an unhandled request arrives.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.

Definition at line `1053` of file `include/openthread/coap.h`

otCoapSendResponseWithParameters

> otError otCoapSendResponseWithParameters (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, const otCoapTxParameters *aTxParameters)

Sends a CoAP response from the server with custom transmission parameters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the CoAP response to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.
[in]	aTxParameters	A pointer to transmission parameters for this response. Use NULL for defaults.

Definition at line `1068` of file `include/openthread/coap.h`

otCoapSendResponseBlockWiseWithParameters

> otError otCoapSendResponseBlockWiseWithParameters (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, const otCoapTxParameters *aTxParameters, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook)

Sends a CoAP response block-wise from the server with custom transmission parameters.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the CoAP response to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.
[in]	aTxParameters	A pointer to transmission parameters for this response. Use NULL for defaults.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.
[in]	aTransmitHook	A pointer to a hook function for outgoing block-wise transfer.

Is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

Definition at line `1091` of file `include/openthread/coap.h`

otCoapSendResponseBlockWise

> static otError otCoapSendResponseBlockWise (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook)

Sends a CoAP response block-wise from the server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the CoAP response to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.
[in]	aTransmitHook	A pointer to a hook function for outgoing block-wise transfer.

Is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

Definition at line `1114` of file `include/openthread/coap.h`

otCoapSendResponse

```
static otError otCoapSendResponse (otInstance *aInstance, otMessage *aMessage, const otMessageInfo
*aMessageInfo)
```

Sends a CoAP response from the server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the CoAP response to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.

Definition at line `1135` of file `include/openthread/coap.h`

Macro Definition Documentation

OT_DEFAULT_COAP_PORT

```
#define OT_DEFAULT_COAP_PORT
```

Value:

```
5683
```

Default CoAP port, as specified in RFC 7252.

Definition at line `59` of file `include/openthread/coap.h`

OT_COAP_DEFAULT_TOKEN_LENGTH

```
#define OT_COAP_DEFAULT_TOKEN_LENGTH
```

Value:

```
2
```

Default token length.

Definition at line `61` of file `include/openthread/coap.h`

OT_COAP_MAX_TOKEN_LENGTH

```
#define OT_COAP_MAX_TOKEN_LENGTH
```

Value:

```
8
```

Max token length as specified (RFC 7252).

Definition at line `63` of file `include/openthread/coap.h`

OT_COAP_MAX_RETRANSMIT

```
#define OT_COAP_MAX_RETRANSMIT
```

Value:

```
20
```

Max retransmit supported by OpenThread.

Definition at line `65` of file `include/openthread/coap.h`

OT_COAP_MIN_ACK_TIMEOUT

```
#define OT_COAP_MIN_ACK_TIMEOUT
```

Value:

```
1000
```

Minimal ACK timeout in milliseconds supported by OpenThread.

Definition at line `67` of file `include/openthread/coap.h`

OT_COAP_CODE

```
#define OT_COAP_CODE
```

Value:

```
(c, d)
```

Helper macro to define CoAP Code values.

Definition at line `85` of file `include/openthread/coap.h`

otCoapOption

Represents a CoAP option.

Public Attributes

uint16_t	mNumber Option Number.
uint16_t	mLength Option Length.

Public Attribute Documentation

mNumber

```
uint16_t otCoapOption::mNumber
```

Option Number.

Definition at line 159 of file include/openthread/coap.h

mLength

```
uint16_t otCoapOption::mLength
```

Option Length.

Definition at line 160 of file include/openthread/coap.h

otCoapOptionIterator

Acts as an iterator for CoAP options.

Public Attributes

const otMessage *	mMessage CoAP message.
otCoapOption	mOption CoAP message option.
uint16_t	mNextOptionOffset Byte offset of next option.

Public Attribute Documentation

mMessage

```
const otMessage* otCoapOptionIterator::mMessage
```

CoAP message.

Definition at line 169 of file include/openthread/coap.h

mOption

```
otCoapOption otCoapOptionIterator::mOption
```

CoAP message option.

Definition at line 170 of file include/openthread/coap.h

mNextOptionOffset

```
uint16_t otCoapOptionIterator::mNextOptionOffset
```

Byte offset of next option.

Definition at line 171 of file include/openthread/coap.h

otCoapResource

Represents a CoAP resource.

Public Attributes

const char *	mUriPath The URI Path string.
otCoapRequestHandler	mHandler The callback for handling a received request.
void *	mContext Application-specific context.
struct otCoapResource *	mNext The next CoAP resource in the list.

Public Attribute Documentation

mUriPath

```
const char* otCoapResource::mUriPath
```

The URI Path string.

Definition at line 417 of file include/openthread/coap.h

mHandler

```
otCoapRequestHandler otCoapResource::mHandler
```

The callback for handling a received request.

Definition at line 418 of file include/openthread/coap.h

mContext

```
void* otCoapResource::mContext
```

Application-specific context.

Definition at line 419 of file include/openthread/coap.h

mNext

```
struct otCoapResource* otCoapResource::mNext
```

The next CoAP resource in the list.

Definition at line 420 of file include/openthread/coap.h

otCoapBlockwiseResource

Represents a CoAP resource with block-wise transfer.

Public Attributes

const char *	mUriPath The URI Path string.
otCoapRequestHandler	mHandler The callback for handling a received request.
otCoapBlockwiseReceiveHook	mReceiveHook The callback for handling incoming block-wise transfer.
otCoapBlockwiseTransmitHook	mTransmitHook The callback for handling outgoing block-wise transfer.
void *	mContext Application-specific context.
struct otCoapBlockwiseResource *	mNext The next CoAP resource in the list.

Public Attribute Documentation

mUriPath

```
const char* otCoapBlockwiseResource::mUriPath
```

The URI Path string.

Definition at line 429 of file include/openthread/coap.h

mHandler

```
otCoapRequestHandler otCoapBlockwiseResource::mHandler
```

The callback for handling a received request.

Definition at line 430 of file include/openthread/coap.h

mReceiveHook

```
otCoapBlockwiseReceiveHook otCoapBlockwiseResource::mReceiveHook
```

The callback for handling incoming block-wise transfer.

This callback is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

Definition at line 436 of file include/openthread/coap.h

mTransmitHook

> otCoapBlockwiseTransmitHook otCoapBlockwiseResource::mTransmitHook

The callback for handling outgoing block-wise transfer.

This callback is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

Definition at line 442 of file include/openthread/coap.h

mContext

> void* otCoapBlockwiseResource::mContext

Application-specific context.

Definition at line 443 of file include/openthread/coap.h

mNext

> struct otCoapBlockwiseResource* otCoapBlockwiseResource::mNext

The next CoAP resource in the list.

Definition at line 444 of file include/openthread/coap.h

otCoapTxParameters

Represents the CoAP transmission parameters.

Note

- mAckTimeout * ((2 ** (mMaxRetransmit + 1)) - 1) * (mAckRandomFactorNumerator / mAckRandomFactorDenominator) must not exceed what can be represented by a uint32_t (0xffffffff). This limitation allows OpenThread to avoid 64-bit arithmetic.

Public Attributes

uint32_t	mAckTimeout Minimum spacing before first retransmission when ACK is not received, in milliseconds (RFC7252 default value is 2000ms).
uint8_t	mAckRandomFactorNumerator Numerator of ACK_RANDOM_FACTOR used to calculate maximum spacing before first retransmission when ACK is not received (RFC7252 default value of ACK_RANDOM_FACTOR is 1.5; must not be decreased below 1).
uint8_t	mAckRandomFactorDenominator Denominator of ACK_RANDOM_FACTOR used to calculate maximum spacing before first retransmission when ACK is not received (RFC7252 default value of ACK_RANDOM_FACTOR is 1.5; must not be decreased below 1).
uint8_t	mMaxRetransmit Maximum number of retransmissions for CoAP Confirmable messages (RFC7252 default value is 4).

Public Attribute Documentation

mAckTimeout

```
uint32_t otCoapTxParameters::mAckTimeout
```

Minimum spacing before first retransmission when ACK is not received, in milliseconds (RFC7252 default value is 2000ms).

Definition at line `462` of file `include/openthread/coap.h`

mAckRandomFactorNumerator

```
uint8_t otCoapTxParameters::mAckRandomFactorNumerator
```

Numerator of ACK_RANDOM_FACTOR used to calculate maximum spacing before first retransmission when ACK is not received (RFC7252 default value of ACK_RANDOM_FACTOR is 1.5; must not be decreased below 1).

Definition at line `469` of file `include/openthread/coap.h`

mAckRandomFactorDenominator

```
uint8_t otCoapTxParameters::mAckRandomFactorDenominator
```

Denominator of ACK_RANDOM_FACTOR used to calculate maximum spacing before first retransmission when ACK is not received (RFC7252 default value of ACK_RANDOM_FACTOR is 1.5; must not be decreased below 1).

Definition at line 476 of file include/openthread/coap.h

mMaxRetransmit

> uint8_t otCoapTxParameters::mMaxRetransmit

Maximum number of retransmissions for CoAP Confirmable messages (RFC7252 default value is 4).

Definition at line 482 of file include/openthread/coap.h

CoAP Secure

CoAP Secure

This module includes functions that control CoAP Secure (CoAP over DTLS) communication.

The functions in this module are available when CoAP Secure API feature (`OPENTHREAD_CONFIG_COAP_SECURE_API_ENABLE`) is enabled.

Typedefs

typedef void(* otHandleCoapSecureClientConnect)(bool aConnected, void *aContext)
Pointer is called when the DTLS connection state changes.

Functions

otError otCoapSecureStart(otInstance *aInstance, uint16_t aPort)
Starts the CoAP Secure service.

void otCoapSecureStop(otInstance *aInstance)
Stops the CoAP Secure server.

void otCoapSecureSetPsk(otInstance *aInstance, const uint8_t *aPsk, uint16_t aPskLength, const uint8_t *aPskIdentity, uint16_t aPskIdLength)
Sets the Pre-Shared Key (PSK) and cipher suite DTLS_PSK_WITH_AES_128_CCM_8.

otError otCoapSecureGetPeerCertificateBase64(otInstance *aInstance, unsigned char *aPeerCert, size_t *aCertLength, size_t aCertBufferSize)
Returns the peer x509 certificate base64 encoded.

void otCoapSecureSetSslAuthMode(otInstance *aInstance, bool aVerifyPeerCertificate)
Sets the authentication mode for the coap secure connection.

void otCoapSecureSetCertificate(otInstance *aInstance, const uint8_t *aX509Cert, uint32_t aX509Length, const uint8_t *aPrivateKey, uint32_t aPrivateKeyLength)
Sets the local device's X509 certificate with corresponding private key for DTLS session with DTLS_ECDHE_ECDSA_WITH_AES_128_CCM_8.

void otCoapSecureSetCaCertificateChain(otInstance *aInstance, const uint8_t *aX509CaCertificateChain, uint32_t aX509CaCertChainLength)
Sets the trusted top level CAs.

otError otCoapSecureConnect(otInstance *aInstance, const otSockAddr *aSockAddr, otHandleCoapSecureClientConnect aHandler, void *aContext)
Initializes DTLS session with a peer.

void otCoapSecureDisconnect(otInstance *aInstance)
Stops the DTLS connection.

bool otCoapSecureIsConnected(otInstance *aInstance)
Indicates whether or not the DTLS session is connected.

bool otCoapSecureIsConnectionActive(otInstance *aInstance)
Indicates whether or not the DTLS session is active.

otError otCoapSecureSendRequestBlockWise(otInstance *aInstance, otMessage *aMessage, otCoapResponseHandler aHandler, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook, otCoapBlockwiseReceiveHook aReceiveHook)
Sends a CoAP request block-wise over secure DTLS connection.

otError otCoapSecureSendRequest(otInstance *aInstance, otMessage *aMessage, otCoapResponseHandler aHandler, void *aContext)
Sends a CoAP request over secure DTLS connection.

void otCoapSecureAddResource(otInstance *aInstance, otCoapResource *aResource)
Adds a resource to the CoAP Secure server.

void otCoapSecureRemoveResource(otInstance *aInstance, otCoapResource *aResource)
Removes a resource from the CoAP Secure server.

void otCoapSecureAddBlockWiseResource(otInstance *aInstance, otCoapBlockwiseResource *aResource)
Adds a block-wise resource to the CoAP Secure server.

void otCoapSecureRemoveBlockWiseResource(otInstance *aInstance, otCoapBlockwiseResource *aResource)
Removes a block-wise resource from the CoAP Secure server.

void otCoapSecureSetDefaultHandler(otInstance *aInstance, otCoapRequestHandler aHandler, void *aContext)
Sets the default handler for unhandled CoAP Secure requests.

void otCoapSecureSetClientConnectedCallback(otInstance *aInstance, otHandleCoapSecureClientConnect aHandler, void *aContext)
Sets the connected callback to indicate, when a Client connect to the CoAP Secure server.

otError otCoapSecureSendResponseBlockWise(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook)
Sends a CoAP response block-wise from the CoAP Secure server.

otError otCoapSecureSendResponse(otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo)
Sends a CoAP response from the CoAP Secure server.

Macros

#define OT_DEFAULT_COAP_SECURE_PORT 5684
Default CoAP Secure port, as specified in RFC 7252.

Typedef Documentation

otHandleCoapSecureClientConnect

```
typedef void(* otHandleCoapSecureClientConnect) (bool aConnected, void *aContext) )(bool aConnected, void *aContext)
```

Pointer is called when the DTLS connection state changes.

Parameters

[in]	aConnected	true, if a connection was established, false otherwise.
[in]	aContext	A pointer to arbitrary context information.

Definition at line 77 of file include/openthread/coap_secure.h

Function Documentation

otCoapSecureStart

> otError otCoapSecureStart (otInstance *aInstance, uint16_t aPort)

Starts the CoAP Secure service.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPort	The local UDP port to bind to.

Definition at line `88` of file `include/openthread/coap_secure.h`

otCoapSecureStop

> void otCoapSecureStop (otInstance *aInstance)

Stops the CoAP Secure server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `96` of file `include/openthread/coap_secure.h`

otCoapSecureSetPsk

> void otCoapSecureSetPsk (otInstance *aInstance, const uint8_t *aPsk, uint16_t aPskLength, const uint8_t *aPskIdentity, uint16_t aPskIdLength)

Sets the Pre-Shared Key (PSK) and cipher suite DTLS_PSK_WITH_AES_128_CCM_8.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aPsk	A pointer to the PSK.
[in]	aPskLength	The PSK length.
[in]	aPskIdentity	The Identity Name for the PSK.
[in]	aPskIdLength	The PSK Identity Length.

Note

- This function requires the build-time feature `MBEDTLS_KEY_EXCHANGE_PSK_ENABLED` to be enabled.

Definition at line `111` of file `include/openthread/coap_secure.h`

otCoapSecureGetPeerCertificateBase64

> otError otCoapSecureGetPeerCertificateBase64 (otInstance *aInstance, unsigned char *aPeerCert, size_t *aCertLength, size_t aCertBufferSize)

Returns the peer x509 certificate base64 encoded.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

[out]	aPeerCert	A pointer to the base64 encoded certificate buffer.
[out]	aCertLength	The length of the base64 encoded peer certificate.
[in]	aCertBufferSize	The buffer size of aPeerCert.

Note

- This function requires the build-time features `MBEDTLS_BASE64_C` and `MBEDTLS_SSL_KEEP_PEER_CERTIFICATE` to be enabled.

Definition at line `133` of file `include/openthread/coap_secure.h`

otCoapSecureSetSslAuthMode

```
void otCoapSecureSetSslAuthMode (otInstance *aInstance, bool aVerifyPeerCertificate)
```

Sets the authentication mode for the coap secure connection.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aVerifyPeerCertificate	true, to verify the peer certificate.

Disable or enable the verification of peer certificate. Must be called before start.

Definition at line `148` of file `include/openthread/coap_secure.h`

otCoapSecureSetCertificate

```
void otCoapSecureSetCertificate (otInstance *aInstance, const uint8_t *aX509Cert, uint32_t aX509Length, const uint8_t *aPrivateKey, uint32_t aPrivateKeyLength)
```

Sets the local device's X509 certificate with corresponding private key for DTLS session with DTLS_ECDHE_ECDSA_WITH_AES_128_CCM_8.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aX509Cert	A pointer to the PEM formatted X509 certificate.
[in]	aX509Length	The length of certificate.
[in]	aPrivateKey	A pointer to the PEM formatted private key.
[in]	aPrivateKeyLength	The length of the private key.

Note

- This function requires `MBEDTLS_KEY_EXCHANGE_ECDHE_ECDSA_ENABLED=1` .

Definition at line `163` of file `include/openthread/coap_secure.h`

otCoapSecureSetCaCertificateChain

```
void otCoapSecureSetCaCertificateChain (otInstance *aInstance, const uint8_t *aX509CaCertificateChain, uint32_t aX509CaCertChainLength)
```

Sets the trusted top level CAs.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aX509CaCertificateChain	A pointer to the PEM formatted X509 CA chain.
[in]	aX509CaCertChainLength	The length of chain.

It is needed for validating the certificate of the peer.

DTLS mode "ECDHE ECDSA with AES 128 CCM 8" for Application CoAPS.

Note

- This function requires `MBEDTLS_KEY_EXCHANGE_ECDHE_ECDSA_ENABLED=1` .

Definition at line `182` of file `include/openthread/coap_secure.h`

otCoapSecureConnect

```
otError otCoapSecureConnect (otInstance *aInstance, const otSockAddr *aSockAddr,
otHandleCoapSecureClientConnect aHandler, void *aContext)
```

Initializes DTLS session with a peer.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aSockAddr	A pointer to the remote socket address.
[in]	aHandler	A pointer to a function that will be called when the DTLS connection state changes.
[in]	aContext	A pointer to arbitrary context information.

Definition at line `198` of file `include/openthread/coap_secure.h`

otCoapSecureDisconnect

```
void otCoapSecureDisconnect (otInstance *aInstance)
```

Stops the DTLS connection.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `209` of file `include/openthread/coap_secure.h`

otCoapSecureIsConnected

```
bool otCoapSecureIsConnected (otInstance *aInstance)
```

Indicates whether or not the DTLS session is connected.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `220` of file `include/openthread/coap_secure.h`

otCoapSecureIsConnectionActive

```
bool otCoapSecureIsConnectionActive (otInstance *aInstance)
```

Indicates whether or not the DTLS session is active.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line 231 of file include/openthread/coap_secure.h

otCoapSecureSendRequestBlockWise

```
otError otCoapSecureSendRequestBlockWise (otInstance *aInstance, otMessage *aMessage, otCoapResponseHandler aHandler, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook, otCoapBlockwiseReceiveHook aReceiveHook)
```

Sends a CoAP request block-wise over secure DTLS connection.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A reference to the message to send.
[in]	aHandler	A function pointer that shall be called on response reception or time-out.
[in]	aContext	A pointer to arbitrary context information.
[in]	aTransmitHook	A function pointer that is called on Block1 response reception.
[in]	aReceiveHook	A function pointer that is called on Block2 response reception.

Is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

If a response for a request is expected, respective function and context information should be provided. If no response is expected, these arguments should be NULL pointers. If Message Id was not set in the header (equal to 0), this function will assign unique Message Id to the message.

Definition at line 255 of file include/openthread/coap_secure.h

otCoapSecureSendRequest

```
otError otCoapSecureSendRequest (otInstance *aInstance, otMessage *aMessage, otCoapResponseHandler aHandler, void *aContext)
```

Sends a CoAP request over secure DTLS connection.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A reference to the message to send.
[in]	aHandler	A function pointer that shall be called on response reception or time-out.
[in]	aContext	A pointer to arbitrary context information.

If a response for a request is expected, respective function and context information should be provided. If no response is expected, these arguments should be NULL pointers. If Message Id was not set in the header (equal to 0), this function will assign unique Message Id to the message.

Definition at line 279 of file include/openthread/coap_secure.h

otCoapSecureAddResource

> void otCoapSecureAddResource (otInstance *aInstance, otCoapResource *aResource)

Adds a resource to the CoAP Secure server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aResource	A pointer to the resource.

Definition at line 291 of file include/openthread/coap_secure.h

otCoapSecureRemoveResource

> void otCoapSecureRemoveResource (otInstance *aInstance, otCoapResource *aResource)

Removes a resource from the CoAP Secure server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aResource	A pointer to the resource.

Definition at line 300 of file include/openthread/coap_secure.h

otCoapSecureAddBlockWiseResource

> void otCoapSecureAddBlockWiseResource (otInstance *aInstance, otCoapBlockwiseResource *aResource)

Adds a block-wise resource to the CoAP Secure server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aResource	A pointer to the resource.

Definition at line 309 of file include/openthread/coap_secure.h

otCoapSecureRemoveBlockWiseResource

> void otCoapSecureRemoveBlockWiseResource (otInstance *aInstance, otCoapBlockwiseResource *aResource)

Removes a block-wise resource from the CoAP Secure server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aResource	A pointer to the resource.

Definition at line 318 of file include/openthread/coap_secure.h

otCoapSecureSetDefaultHandler

> void otCoapSecureSetDefaultHandler (otInstance *aInstance, otCoapRequestHandler aHandler, void *aContext)

Sets the default handler for unhandled CoAP Secure requests.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aHandler	A function pointer that shall be called when an unhandled request arrives.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.

Definition at line `328` of file `include/openthread/coap_secure.h`

otCoapSecureSetClientConnectedCallback

> void otCoapSecureSetClientConnectedCallback (otInstance *aInstance, otHandleCoapSecureClientConnect aHandler, void *aContext)

Sets the connected callback to indicate, when a Client connect to the CoAP Secure server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aHandler	A pointer to a function that will be called once DTLS connection is established.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.

Definition at line `339` of file `include/openthread/coap_secure.h`

otCoapSecureSendResponseBlockWise

> otError otCoapSecureSendResponseBlockWise (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo, void *aContext, otCoapBlockwiseTransmitHook aTransmitHook)

Sends a CoAP response block-wise from the CoAP Secure server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the CoAP response to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.
[in]	aContext	A pointer to arbitrary context information. May be NULL if not used.
[in]	aTransmitHook	A function pointer that is called on Block1 request reception.

Is available when OPENTHREAD_CONFIG_COAP_BLOCKWISE_TRANSFER_ENABLE configuration is enabled.

Definition at line `359` of file `include/openthread/coap_secure.h`

otCoapSecureSendResponse

> otError otCoapSecureSendResponse (otInstance *aInstance, otMessage *aMessage, const otMessageInfo *aMessageInfo)

Sends a CoAP response from the CoAP Secure server.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMessage	A pointer to the CoAP response to send.
[in]	aMessageInfo	A pointer to the message info associated with `aMessage`.

Definition at line `376` of file `include/openthread/coap_secure.h`

Macro Definition Documentation

OT_DEFAULT_COAP_SECURE_PORT

```
#define OT_DEFAULT_COAP_SECURE_PORT
```

Value:

```
5684
```

Default CoAP Secure port, as specified in RFC 7252.

Definition at line `68` of file `include/openthread/coap_secure.h`

Command Line Interface

Command Line Interface

This module includes functions that control the Thread stack's execution.

Modules

otCliCommand

Typedefs

typedef int(* otCliOutputCallback)(void *aContext, const char *aFormat, va_list aArguments)
Pointer is called to notify about Console output.

typedef struct otCliCommand otCliCommand

Functions

void otCliInit(otInstance *aInstance, otCliOutputCallback aCallback, void *aContext)
Initialize the CLI module.

void otCliInputLine(char *aBuf)
Is called to feed in a console input line.

otError otCliSetUserCommands(const otCliCommand *aUserCommands, uint8_t aLength, void *aContext)
Set a user command table.

void otCliOutputBytes(const uint8_t *aBytes, uint8_t aLength)
Write a number of bytes to the CLI console as a hex string.

void otCliOutputFormat(const char *aFmt,...)
Write formatted string to the CLI console.

void otCliAppendResult(otError aError)
Write error code to the CLI console.

void otCliPlatLogv(otLogLevel aLogLevel, otLogRegion aLogRegion, const char *aFormat, va_list aArgs)
Callback to write the OpenThread Log to the CLI console.

void otCliVendorSetUserCommands(void)
Callback to allow vendor specific commands to be added to the user command table.

Typedef Documentation

otCliOutputCallback

```
typedef int(* otCliOutputCallback) (void *aContext, const char *aFormat, va_list aArguments) )(void *aContext, const char *aFormat, va_list aArguments)
```

Pointer is called to notify about Console output.

Parameters

[out]	aContext	A user context pointer.
[in]	aFormat	The format string.
[in]	aArguments	The format string arguments.

Returns

- Number of bytes written by the callback.

Definition at line `80` of file `include/openthread/cli.h`

otCliCommand

```
typedef struct otCliCommand otCliCommand
```

Definition at line `10` of file `doxygen_overrides.txt`

Function Documentation

otCliInit

```
void otCliInit (otInstance *aInstance, otCliOutputCallback aCallback, void *aContext)
```

Initialize the CLI module.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aCallback	A callback method called to process CLI output.
[in]	aContext	A user context pointer.

Definition at line `90` of file `include/openthread/cli.h`

otCliInputLine

```
void otCliInputLine (char *aBuf)
```

Is called to feed in a console input line.

Parameters

[in]	aBuf	A pointer to a null-terminated string.

Definition at line `98` of file `include/openthread/cli.h`

otCliSetUserCommands

```
otError otCliSetUserCommands (const otCliCommand *aUserCommands, uint8_t aLength, void *aContext)
```

Set a user command table.

Parameters

[in]	aUserCommands	A pointer to an array with user commands.
[in]	aLength	`aUserCommands` length.
[in]	aContext	`The` context passed to the handler.

Definition at line `110` of file `include/openthread/cli.h`

otCliOutputBytes

> void otCliOutputBytes (const uint8_t *aBytes, uint8_t aLength)

Write a number of bytes to the CLI console as a hex string.

Parameters

[in]	aBytes	A pointer to data which should be printed.
[in]	aLength	`aBytes` length.

Definition at line `119` of file `include/openthread/cli.h`

otCliOutputFormat

> void otCliOutputFormat (const char *aFmt,...)

Write formatted string to the CLI console.

Parameters

[in]	aFmt	A pointer to the format string.
[in]		A matching list of arguments.

Definition at line `128` of file `include/openthread/cli.h`

otCliAppendResult

> void otCliAppendResult (otError aError)

Write error code to the CLI console.

Parameters

[in]	aError	Error code value.

If the `aError` is `OT_ERROR_PENDING` nothing will be outputted.

Definition at line `138` of file `include/openthread/cli.h`

otCliPlatLogv

> void otCliPlatLogv (otLogLevel aLogLevel, otLogRegion aLogRegion, const char *aFormat, va_list aArgs)

Callback to write the OpenThread Log to the CLI console.

Parameters

[in]	aLogLevel	The log level.
[in]	aLogRegion	The log region.
[in]	aFormat	A pointer to the format string.
[in]	aArgs	va_list matching aFormat.

Definition at line `149` of file `include/openthread/cli.h`

otCliVendorSetUserCommands

```
void otCliVendorSetUserCommands (void)
```

Callback to allow vendor specific commands to be added to the user command table.

Parameters

N/A		

Available when `OPENTHREAD_CONFIG_CLI_VENDOR_COMMANDS_ENABLE` is enabled and `OPENTHREAD_CONFIG_CLI_MAX_USER_CMD_ENTRIES` is greater than 1.

Definition at line `158` of file `include/openthread/cli.h`

otCliCommand

Represents a CLI command.

Public Attributes

const char *	mName A pointer to the command string.
otError(*	mCommand A function pointer to process the command.
void(*	mCommand A function pointer to process the command.

Public Attribute Documentation

mName

```
const char * otCliCommand::mName
```

A pointer to the command string.

Definition at line 54 of file include/openthread/cli.h

mCommand

```
otError(* otCliCommand::mCommand) (void *aContext, uint8_t aArgsLength, char *aArgs[])
```

A function pointer to process the command.

Definition at line 55 of file include/openthread/cli.h

mCommand

```
void(* otCliCommand::mCommand) (void *aContext, uint8_t aArgsLength, char *aArgs[])
```

A function pointer to process the command.

Definition at line 7 of file doxygen_overrides.txt

Crypto - Thread Stack

This module includes cryptographic functions.

Typedefs

typedef otPlatCryptoSha256Hash	otCryptoSha256Hash Represents a SHA-256 hash.

Functions

void	otCryptoHmacSha256(const otCryptoKey *aKey, const uint8_t *aBuf, uint16_t aBufLength, otCryptoSha256Hash *aHash) Performs HMAC computation.
void	otCryptoAesCcm(const otCryptoKey *aKey, uint8_t aTagLength, const void *aNonce, uint8_t aNonceLength, const void *aHeader, uint32_t aHeaderLength, void *aPlainText, void *aCipherText, uint32_t aLength, bool aEncrypt, void *aTag) Performs AES CCM computation.

Typedef Documentation

otCryptoSha256Hash

```
typedef otPlatCryptoSha256Hash otCryptoSha256Hash
```

Represents a SHA-256 hash.

Definition at line `62` of file `include/openthread/crypto.h`

Function Documentation

otCryptoHmacSha256

```
void otCryptoHmacSha256 (const otCryptoKey *aKey, const uint8_t *aBuf, uint16_t aBufLength, otCryptoSha256Hash *aHash)
```

Performs HMAC computation.

Parameters

[in]	aKey	A pointer to the key.
[in]	aBuf	A pointer to the input buffer.
[in]	aBufLength	The length of `aBuf` in bytes.
[out]	aHash	A pointer to a `otCryptoSha256Hash` structure to output the hash value.

Definition at line 73 of file include/openthread/crypto.h

otCryptoAesCcm

> void otCryptoAesCcm (const otCryptoKey *aKey, uint8_t aTagLength, const void *aNonce, uint8_t aNonceLength, const void *aHeader, uint32_t aHeaderLength, void *aPlainText, void *aCipherText, uint32_t aLength, bool aEncrypt, void *aTag)

Performs AES CCM computation.

Parameters

[in]	aKey	A pointer to the key.
[in]	aTagLength	Length of tag in bytes.
[in]	aNonce	A pointer to the nonce.
[in]	aNonceLength	Length of nonce in bytes.
[in]	aHeader	A pointer to the header.
[in]	aHeaderLength	Length of header in bytes.
[inout]	aPlainText	A pointer to the plaintext.
[inout]	aCipherText	A pointer to the ciphertext.
[in]	aLength	Plaintext length in bytes.
[in]	aEncrypt	true on encrypt and false on decrypt.
[out]	aTag	A pointer to the tag.

Definition at line 94 of file include/openthread/crypto.h

Factory Diagnostics - Thread Stack

This module includes functions that control the Thread stack's execution.

Functions

otError otDiagProcessCmd(otInstance *aInstance, uint8_t aArgsLength, char *aArgs[], char *aOutput, size_t aOutputMaxLen)
Processes a factory diagnostics command line.

otError otDiagProcessCmdLine(otInstance *aInstance, const char *aString, char *aOutput, size_t aOutputMaxLen)
Processes a factory diagnostics command line.

bool otDiagIsEnabled(otInstance *aInstance)
Indicates whether or not the factory diagnostics mode is enabled.

Function Documentation

otDiagProcessCmd

> otError otDiagProcessCmd (otInstance *aInstance, uint8_t aArgsLength, char *aArgs[], char *aOutput, size_t aOutputMaxLen)

Processes a factory diagnostics command line.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aArgsLength	The number of elements in `aArgs`.
[in]	aArgs	An array of arguments.
[out]	aOutput	The diagnostics execution result.
[in]	aOutputMaxLen	The output buffer size.

The output of this function (the content written to `aOutput`) MUST terminate with `\0` and the `\0` is within the output buffer.

Definition at line `71` of file `include/openthread/diag.h`

otDiagProcessCmdLine

> otError otDiagProcessCmdLine (otInstance *aInstance, const char *aString, char *aOutput, size_t aOutputMaxLen)

Processes a factory diagnostics command line.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aString	A NULL-terminated input string.

[out]	aOutput	The diagnostics execution result.
[in]	aOutputMaxLen	The output buffer size.

The output of this function (the content written to `aOutput`) MUST terminate with `\0` and the `\0` is within the output buffer.

Definition at line `94` of file `include/openthread/diag.h`

otDiagIsEnabled

```
bool otDiagIsEnabled (otInstance *aInstance)
```

Indicates whether or not the factory diagnostics mode is enabled.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `105` of file `include/openthread/diag.h`

Heap

Heap

This module includes functions that set the external OpenThread heap.

Functions

void *	otHeapCAlloc(size_t aCount, size_t aSize)
void	otHeapFree(void *aPointer)

Function Documentation

otHeapCAlloc

```
void * otHeapCAlloc (size_t aCount, size_t aSize)
```

Parameters

N/A	aCount	
N/A	aSize	

Note

- This API is deprecated and use of it is discouraged.

Definition at line `60` of file `include/openthread/heap.h`

otHeapFree

```
void otHeapFree (void *aPointer)
```

Parameters

N/A	aPointer	

Note

- This API is deprecated and use of it is discouraged.

Definition at line `68` of file `include/openthread/heap.h`

History Tracker

History Tracker

Records the history of different events, for example RX and TX messages or network info changes.

All tracked entries are timestamped.

The functions in this module are available when `OPENTHREAD_CONFIG_HISTORY_TRACKER_ENABLE` is enabled.

Modules

otHistoryTrackerIterator

otHistoryTrackerNetworkInfo

otHistoryTrackerUnicastAddressInfo

otHistoryTrackerMulticastAddressInfo

otHistoryTrackerMessageInfo

otHistoryTrackerNeighborInfo

otHistoryTrackerRouterInfo

otHistoryTrackerOnMeshPrefixInfo

otHistoryTrackerExternalRouteInfo

Enumerations

enum otHistoryTrackerAddressEvent {

```
OT_HISTORY_TRACKER_ADDRESS_EVENT_ADDED = 0
OT_HISTORY_TRACKER_ADDRESS_EVENT_REMOVED = 1
```

}

Defines the events for an IPv6 (unicast or multicast) address info (i.e., whether address is added or removed).

enum @1 {

```
OT_HISTORY_TRACKER_MSG_PRIORITY_LOW = OT_MESSAGE_PRIORITY_LOW
OT_HISTORY_TRACKER_MSG_PRIORITY_NORMAL = OT_MESSAGE_PRIORITY_NORMAL
OT_HISTORY_TRACKER_MSG_PRIORITY_HIGH = OT_MESSAGE_PRIORITY_HIGH
OT_HISTORY_TRACKER_MSG_PRIORITY_NET = OT_MESSAGE_PRIORITY_HIGH + 1
```

}

Constants representing message priority used in `otHistoryTrackerMessageInfo` struct.

enum otHistoryTrackerNeighborEvent {

```
OT_HISTORY_TRACKER_NEIGHBOR_EVENT_ADDED = 0
OT_HISTORY_TRACKER_NEIGHBOR_EVENT_REMOVED = 1
OT_HISTORY_TRACKER_NEIGHBOR_EVENT_CHANGED = 2
OT_HISTORY_TRACKER_NEIGHBOR_EVENT_RESTORING = 3
```

}

Defines the events in a neighbor info (i.e.

enum otHistoryTrackerRouterEvent {

```
OT_HISTORY_TRACKER_ROUTER_EVENT_ADDED = 0
OT_HISTORY_TRACKER_ROUTER_EVENT_REMOVED = 1
OT_HISTORY_TRACKER_ROUTER_EVENT_NEXT_HOP_CHANGED = 2
OT_HISTORY_TRACKER_ROUTER_EVENT_COST_CHANGED = 3
```

}

Defines the events in a router info (i.e.

enum otHistoryTrackerNetDataEvent {

```
OT_HISTORY_TRACKER_NET_DATA_ENTRY_ADDED = 0
OT_HISTORY_TRACKER_NET_DATA_ENTRY_REMOVED = 1
```

}

Defines the events for a Network Data entry (i.e., whether an entry is added or removed).

Typedefs

typedef struct otHistoryTrackerIterator — Represents an iterator to iterate through a history list.

typedef struct otHistoryTrackerNetworkInfo — Represents Thread network info.

typedef struct otHistoryTrackerUnicastAddressInfo — Represent a unicast IPv6 address info.

typedef struct otHistoryTrackerMulticastAddressInfo — Represent an IPv6 multicast address info.

typedef struct otHistoryTrackerMessageInfo — Represents a RX/TX IPv6 message info.

typedef struct otHistoryTrackerNeighborInfo — Represents a neighbor info.

typedef struct otHistoryTrackerRouterInfo — Represents a router table entry event.

typedef struct otHistoryTrackerOnMeshPrefixInfo — Represent a Network Data on mesh prefix info.

typedef struct otHistoryTrackerExternalRouteInfo — Represent a Network Data extern route info.

Functions

void otHistoryTrackerInitIterator(otHistoryTrackerIterator *aIterator)

Initializes an `otHistoryTrackerIterator` .

const otHistoryTrackerNetworkInfo *	otHistoryTrackerIterateNetInfoHistory(otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge) Iterates over the entries in the network info history list.
const otHistoryTrackerUnicastAddressInfo *	otHistoryTrackerIterateUnicastAddressHistory(otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge) Iterates over the entries in the unicast address history list.
const otHistoryTrackerMulticastAddressInfo *	otHistoryTrackerIterateMulticastAddressHistory(otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge) Iterates over the entries in the multicast address history list.
const otHistoryTrackerMessageInfo *	otHistoryTrackerIterateRxHistory(otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge) Iterates over the entries in the RX message history list.
const otHistoryTrackerMessageInfo *	otHistoryTrackerIterateTxHistory(otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge) Iterates over the entries in the TX message history list.
const otHistoryTrackerNeighborInfo *	otHistoryTrackerIterateNeighborHistory(otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge) Iterates over the entries in the neighbor history list.
const otHistoryTrackerRouterInfo *	otHistoryTrackerIterateRouterHistory(otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge) Iterates over the entries in the router history list.
const otHistoryTrackerOnMeshPrefixInfo *	otHistoryTrackerIterateOnMeshPrefixHistory(otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge) Iterates over the entries in the Network Data on mesh prefix entry history list.
const otHistoryTrackerExternalRouteInfo *	otHistoryTrackerIterateExternalRouteHistory(otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge) Iterates over the entries in the Network Data external route entry history list.
void	otHistoryTrackerEntryAgeToString(uint32_t aEntryAge, char *aBuffer, uint16_t aSize) Converts a given entry age to a human-readable string.

Macros

#define	OT_HISTORY_TRACKER_MAX_AGE (49 * 24 * 60 * 60 * 1000u) This constant specifies the maximum age of entries which is 49 days (in msec).
#define	OT_HISTORY_TRACKER_ENTRY_AGE_STRING_SIZE 21 Recommended size for string representation of an entry age.
#define	OT_HISTORY_TRACKER_NO_NEXT_HOP 63 No next hop - For `mNextHop` in `otHistoryTrackerRouterInfo`.
#define	OT_HISTORY_TRACKER_INFINITE_PATH_COST 0 Infinite path cost - used in `otHistoryTrackerRouterInfo`.

Enumeration Documentation

otHistoryTrackerAddressEvent

> otHistoryTrackerAddressEvent

Defines the events for an IPv6 (unicast or multicast) address info (i.e., whether address is added or removed).

Enumerator	
OT_HISTORY_TRACKER_ADDRESS_EVENT_ADDED	Address is added.
OT_HISTORY_TRACKER_ADDRESS_EVENT_REMOVED	Address is removed.

Definition at line `94` of file `include/openthread/history_tracker.h`

@1

> @1

Constants representing message priority used in `otHistoryTrackerMessageInfo` struct.

Enumerator	
OT_HISTORY_TRACKER_MSG_PRIORITY_LOW	Low priority level.
OT_HISTORY_TRACKER_MSG_PRIORITY_NORMAL	Normal priority level.
OT_HISTORY_TRACKER_MSG_PRIORITY_HIGH	High priority level.
OT_HISTORY_TRACKER_MSG_PRIORITY_NET	Network Control priority level.

Definition at line `131` of file `include/openthread/history_tracker.h`

otHistoryTrackerNeighborEvent

> otHistoryTrackerNeighborEvent

Defines the events in a neighbor info (i.e.

whether neighbor is added, removed, or changed).

Event `OT_HISTORY_TRACKER_NEIGHBOR_EVENT_RESTORING` is applicable to child neighbors only. It is triggered after the device (re)starts and when the previous children list is retrieved from non-volatile settings and the device tries to restore connection to them.

Enumerator	
OT_HISTORY_TRACKER_NEIGHBOR_EVENT_ADDED	Neighbor is added.
OT_HISTORY_TRACKER_NEIGHBOR_EVENT_REMOVED	Neighbor is removed.
OT_HISTORY_TRACKER_NEIGHBOR_EVENT_CHANGED	Neighbor changed (e.g., device mode flags changed).
OT_HISTORY_TRACKER_NEIGHBOR_EVENT_RESTORING	Neighbor is being restored (applicable to child only).

Definition at line `171` of file `include/openthread/history_tracker.h`

otHistoryTrackerRouterEvent

> otHistoryTrackerRouterEvent

Defines the events in a router info (i.e.

whether router is added, removed, or changed).

Enumerator	
OT_HISTORY_TRACKER_ROUTER_EVENT_ADDED	Router is added (router ID allocated).
OT_HISTORY_TRACKER_ROUTER_EVENT_REMOVED	Router entry is removed (router ID released).
OT_HISTORY_TRACKER_ROUTER_EVENT_NEXT_HOP_CHANGED	Router entry next hop and cost changed.
OT_HISTORY_TRACKER_ROUTER_EVENT_COST_CHANGED	Router entry path cost changed (next hop as before).

Definition at line `199` of file `include/openthread/history_tracker.h`

otHistoryTrackerNetDataEvent

> otHistoryTrackerNetDataEvent

Defines the events for a Network Data entry (i.e., whether an entry is added or removed).

Enumerator	
OT_HISTORY_TRACKER_NET_DATA_ENTRY_ADDED	Network data entry is added.
OT_HISTORY_TRACKER_NET_DATA_ENTRY_REMOVED	Network data entry is removed.

Definition at line `228` of file `include/openthread/history_tracker.h`

Typedef Documentation

otHistoryTrackerIterator

> typedef struct otHistoryTrackerIterator otHistoryTrackerIterator

Represents an iterator to iterate through a history list.

The fields in this type are opaque (intended for use by OpenThread core) and therefore should not be accessed/used by caller.

Before using an iterator, it MUST be initialized using `otHistoryTrackerInitIterator()` ,

Definition at line `75` of file `include/openthread/history_tracker.h`

otHistoryTrackerNetworkInfo

> typedef struct otHistoryTrackerNetworkInfo otHistoryTrackerNetworkInfo

Represents Thread network info.

Definition at line `87` of file `include/openthread/history_tracker.h`

otHistoryTrackerUnicastAddressInfo

> typedef struct otHistoryTrackerUnicastAddressInfo otHistoryTrackerUnicastAddressInfo

Represent a unicast IPv6 address info.

Definition at line `114` of file `include/openthread/history_tracker.h`

otHistoryTrackerMulticastAddressInfo

```
typedef struct otHistoryTrackerMulticastAddressInfo otHistoryTrackerMulticastAddressInfo
```

Represent an IPv6 multicast address info.

Definition at line `125` of file `include/openthread/history_tracker.h`

otHistoryTrackerMessageInfo

```
typedef struct otHistoryTrackerMessageInfo otHistoryTrackerMessageInfo
```

Represents a RX/TX IPv6 message info.

Some of the fields in this struct are applicable to a RX message or a TX message only, e.g., `mAveRxRss` is the average RSS of all fragment frames that form a received message and is only applicable for a RX message.

Definition at line `161` of file `include/openthread/history_tracker.h`

otHistoryTrackerNeighborInfo

```
typedef struct otHistoryTrackerNeighborInfo otHistoryTrackerNeighborInfo
```

Represents a neighbor info.

Definition at line `193` of file `include/openthread/history_tracker.h`

otHistoryTrackerRouterInfo

```
typedef struct otHistoryTrackerRouterInfo otHistoryTrackerRouterInfo
```

Represents a router table entry event.

Definition at line `222` of file `include/openthread/history_tracker.h`

otHistoryTrackerOnMeshPrefixInfo

```
typedef struct otHistoryTrackerOnMeshPrefixInfo otHistoryTrackerOnMeshPrefixInfo
```

Represent a Network Data on mesh prefix info.

Definition at line `242` of file `include/openthread/history_tracker.h`

otHistoryTrackerExternalRouteInfo

```
typedef struct otHistoryTrackerExternalRouteInfo otHistoryTrackerExternalRouteInfo
```

Represent a Network Data extern route info.

Definition at line 252 of file include/openthread/history_tracker.h

Function Documentation

otHistoryTrackerInitIterator

> void otHistoryTrackerInitIterator (otHistoryTrackerIterator *aIterator)

Initializes an otHistoryTrackerIterator .

Parameters

[in]	aIterator	A pointer to the iterator to initialize (MUST NOT be NULL).

An iterator MUST be initialized before it is used.

An iterator can be initialized again to start from the beginning of the list.

When iterating over entries in a list, to ensure the entry ages are consistent, the age is given relative to the time the iterator was initialized, i.e., the entry age is provided as the duration (in milliseconds) from the event (when entry was recorded) to the iterator initialization time.

Definition at line 268 of file include/openthread/history_tracker.h

otHistoryTrackerIterateNetInfoHistory

> const otHistoryTrackerNetworkInfo * otHistoryTrackerIterateNetInfoHistory (otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge)

Iterates over the entries in the network info history list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[inout]	aIterator	A pointer to an iterator. MUST be initialized or the behavior is undefined.
[out]	aEntryAge	A pointer to a variable to output the entry's age. MUST NOT be NULL. Age is provided as the duration (in milliseconds) from when entry was recorded to aIterator initialization time. It is set to OT_HISTORY_TRACKER_MAX_AGE for entries older than max age.

Returns

- A pointer to otHistoryTrackerNetworkInfo entry or NULL if no more entries in the list.

Definition at line 283 of file include/openthread/history_tracker.h

otHistoryTrackerIterateUnicastAddressHistory

> const otHistoryTrackerUnicastAddressInfo * otHistoryTrackerIterateUnicastAddressHistory (otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge)

Iterates over the entries in the unicast address history list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[inout]	aIterator	A pointer to an iterator. MUST be initialized or the behavior is undefined.

[out]	aEntryAge	A pointer to a variable to output the entry's age. MUST NOT be NULL. Age is provided as the duration (in milliseconds) from when entry was recorded to `aIterator` initialization time. It is set to `OT_HISTORY_TRACKER_MAX_AGE` for entries older than max age.

Returns

- A pointer to `otHistoryTrackerUnicastAddressInfo` entry or `NULL` if no more entries in the list.

Definition at line `300` of file `include/openthread/history_tracker.h`

otHistoryTrackerIterateMulticastAddressHistory

> const otHistoryTrackerMulticastAddressInfo * otHistoryTrackerIterateMulticastAddressHistory (otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge)

Iterates over the entries in the multicast address history list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[inout]	aIterator	A pointer to an iterator. MUST be initialized or the behavior is undefined.
[out]	aEntryAge	A pointer to a variable to output the entry's age. MUST NOT be NULL. Age is provided as the duration (in milliseconds) from when entry was recorded to `aIterator` initialization time. It is set to `OT_HISTORY_TRACKER_MAX_AGE` for entries older than max age.

Returns

- A pointer to `otHistoryTrackerMulticastAddressInfo` entry or `NULL` if no more entries in the list.

Definition at line `318` of file `include/openthread/history_tracker.h`

otHistoryTrackerIterateRxHistory

> const otHistoryTrackerMessageInfo * otHistoryTrackerIterateRxHistory (otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge)

Iterates over the entries in the RX message history list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[inout]	aIterator	A pointer to an iterator. MUST be initialized or the behavior is undefined.
[out]	aEntryAge	A pointer to a variable to output the entry's age. MUST NOT be NULL. Age is provided as the duration (in milliseconds) from when entry was recorded to `aIterator` initialization time. It is set to `OT_HISTORY_TRACKER_MAX_AGE` for entries older than max age.

Returns

- The `otHistoryTrackerMessageInfo` entry or `NULL` if no more entries in the list.

Definition at line `336` of file `include/openthread/history_tracker.h`

otHistoryTrackerIterateTxHistory

const otHistoryTrackerMessageInfo * otHistoryTrackerIterateTxHistory (otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge)

Iterates over the entries in the TX message history list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[inout]	aIterator	A pointer to an iterator. MUST be initialized or the behavior is undefined.
[out]	aEntryAge	A pointer to a variable to output the entry's age. MUST NOT be NULL. Age is provided as the duration (in milliseconds) from when entry was recorded to `aIterator` initialization time. It is set to `OT_HISTORY_TRACKER_MAX_AGE` for entries older than max age.

Returns

- The `otHistoryTrackerMessageInfo` entry or `NULL` if no more entries in the list.

Definition at line `353` of file `include/openthread/history_tracker.h`

otHistoryTrackerIterateNeighborHistory

const otHistoryTrackerNeighborInfo * otHistoryTrackerIterateNeighborHistory (otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge)

Iterates over the entries in the neighbor history list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[inout]	aIterator	A pointer to an iterator. MUST be initialized or the behavior is undefined.
[out]	aEntryAge	A pointer to a variable to output the entry's age. MUST NOT be NULL. Age is provided as the duration (in milliseconds) from when entry was recorded to `aIterator` initialization time. It is set to `OT_HISTORY_TRACKER_MAX_AGE` for entries older than max age.

Returns

- The `otHistoryTrackerNeighborInfo` entry or `NULL` if no more entries in the list.

Definition at line `370` of file `include/openthread/history_tracker.h`

otHistoryTrackerIterateRouterHistory

const otHistoryTrackerRouterInfo * otHistoryTrackerIterateRouterHistory (otInstance *aInstance, otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge)

Iterates over the entries in the router history list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[inout]	aIterator	A pointer to an iterator. MUST be initialized or the behavior is undefined.
[out]	aEntryAge	A pointer to a variable to output the entry's age. MUST NOT be NULL. Age is provided as the duration (in milliseconds) from when entry was recorded to `aIterator` initialization time. It is set to `OT_HISTORY_TRACKER_MAX_AGE` for entries older than max age.

Returns

- The `otHistoryTrackerRouterInfo` entry or `NULL` if no more entries in the list.

Definition at line `387` of file `include/openthread/history_tracker.h`

otHistoryTrackerIterateOnMeshPrefixHistory

```
const otHistoryTrackerOnMeshPrefixInfo * otHistoryTrackerIterateOnMeshPrefixHistory (otInstance *aInstance,
otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge)
```

Iterates over the entries in the Network Data on mesh prefix entry history list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[inout]	aIterator	A pointer to an iterator. MUST be initialized or the behavior is undefined.
[out]	aEntryAge	A pointer to a variable to output the entry's age. MUST NOT be NULL. Age is provided as the duration (in milliseconds) from when entry was recorded to `aIterator` initialization time. It is set to `OT_HISTORY_TRACKER_MAX_AGE` for entries older than max age.

Returns

- The `otHistoryTrackerOnMeshPrefixInfo` entry or `NULL` if no more entries in the list.

Definition at line `404` of file `include/openthread/history_tracker.h`

otHistoryTrackerIterateExternalRouteHistory

```
const otHistoryTrackerExternalRouteInfo * otHistoryTrackerIterateExternalRouteHistory (otInstance *aInstance,
otHistoryTrackerIterator *aIterator, uint32_t *aEntryAge)
```

Iterates over the entries in the Network Data external route entry history list.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[inout]	aIterator	A pointer to an iterator. MUST be initialized or the behavior is undefined.
[out]	aEntryAge	A pointer to a variable to output the entry's age. MUST NOT be NULL. Age is provided as the duration (in milliseconds) from when entry was recorded to `aIterator` initialization time. It is set to `OT_HISTORY_TRACKER_MAX_AGE` for entries older than max age.

Returns

- The `otHistoryTrackerExternalRouteInfo` entry or `NULL` if no more entries in the list.

Definition at line `421` of file `include/openthread/history_tracker.h`

otHistoryTrackerEntryAgeToString

```
void otHistoryTrackerEntryAgeToString (uint32_t aEntryAge, char *aBuffer, uint16_t aSize)
```

Converts a given entry age to a human-readable string.

Parameters

[in]	aEntryAge	The entry age (duration in msec).
[out]	aBuffer	A pointer to a char array to output the string (MUST NOT be NULL).
[in]	aSize	The size of `aBuffer`. Recommended to use `OT_HISTORY_TRACKER_ENTRY_AGE_STRING_SIZE`.

The entry age string follows the format "<hh>:<mm>:<ss>.<mmmm>" for hours, minutes, seconds and millisecond (if shorter than one day) or "<dd> days <hh>:<mm>:<ss>.<mmmm>" (if longer than one day).

If the resulting string does not fit in `aBuffer` (within its `aSize` characters), the string will be truncated but the outputted string is always null-terminated.

Definition at line `440` of file `include/openthread/history_tracker.h`

Macro Definition Documentation

OT_HISTORY_TRACKER_MAX_AGE

```
#define OT_HISTORY_TRACKER_MAX_AGE
```

Value:

```
(49 * 24 * 60 * 60 * 1000u)
```

This constant specifies the maximum age of entries which is 49 days (in msec).

Entries older than the max age will give this value as their age.

Definition at line `58` of file `include/openthread/history_tracker.h`

OT_HISTORY_TRACKER_ENTRY_AGE_STRING_SIZE

```
#define OT_HISTORY_TRACKER_ENTRY_AGE_STRING_SIZE
```

Value:

```
21
```

Recommended size for string representation of an entry age.

Definition at line `60` of file `include/openthread/history_tracker.h`

OT_HISTORY_TRACKER_NO_NEXT_HOP

```
#define OT_HISTORY_TRACKER_NO_NEXT_HOP
```

Value:

```
63
```

No next hop - For `mNextHop` in `otHistoryTrackerRouterInfo`.

Definition at line `207` of file `include/openthread/history_tracker.h`

OT_HISTORY_TRACKER_INFINITE_PATH_COST

```
#define OT_HISTORY_TRACKER_INFINITE_PATH_COST
```

Value:

```
0
```

Infinite path cost - used in `otHistoryTrackerRouterInfo`.

Definition at line `209` of file `include/openthread/history_tracker.h`

otHistoryTrackerIterator

Represents an iterator to iterate through a history list.

The fields in this type are opaque (intended for use by OpenThread core) and therefore should not be accessed/used by caller.

Before using an iterator, it MUST be initialized using `otHistoryTrackerInitIterator()` ,

Public Attributes

uint32_t	mData32
uint16_t	mData16

Public Attribute Documentation

mData32

```
uint32_t otHistoryTrackerIterator::mData32
```

Definition at line `73` of file `include/openthread/history_tracker.h`

mData16

```
uint16_t otHistoryTrackerIterator::mData16
```

Definition at line `74` of file `include/openthread/history_tracker.h`

otHistoryTrackerNetworkInfo

Represents Thread network info.

Public Attributes

otDeviceRole	mRole Device Role.
otLinkModeConfig	mMode Device Mode.
uint16_t	mRloc16 Device RLOC16.
uint32_t	mPartitionId Partition ID (valid when attached).

Public Attribute Documentation

mRole

```
otDeviceRole otHistoryTrackerNetworkInfo::mRole
```

Device Role.

Definition at line `83` of file `include/openthread/history_tracker.h`

mMode

```
otLinkModeConfig otHistoryTrackerNetworkInfo::mMode
```

Device Mode.

Definition at line `84` of file `include/openthread/history_tracker.h`

mRloc16

```
uint16_t otHistoryTrackerNetworkInfo::mRloc16
```

Device RLOC16.

Definition at line `85` of file `include/openthread/history_tracker.h`

mPartitionId

```
uint32_t otHistoryTrackerNetworkInfo::mPartitionId
```

Partition ID (valid when attached).

Definition at line 86 of file include/openthread/history_tracker.h

otHistoryTrackerUnicastAddressInfo

Represent a unicast IPv6 address info.

Public Attributes

otIp6Address	mAddress The unicast IPv6 address.
uint8_t	mPrefixLength The Prefix length (in bits).
uint8_t	mAddressOrigin The address origin (`OT_ADDRESS_ORIGIN_*` constants).
otHistoryTrackerAddressEvent	mEvent Indicates the event (address is added/removed).
uint8_t	mScope The IPv6 scope.
bool	mPreferred If the address is preferred.
bool	mValid If the address is valid.
bool	mRloc If the address is an RLOC.

Public Attribute Documentation

mAddress

> otIp6Address otHistoryTrackerUnicastAddressInfo::mAddress

The unicast IPv6 address.

Definition at line `106` of file `include/openthread/history_tracker.h`

mPrefixLength

> uint8_t otHistoryTrackerUnicastAddressInfo::mPrefixLength

The Prefix length (in bits).

Definition at line `107` of file `include/openthread/history_tracker.h`

mAddressOrigin

> uint8_t otHistoryTrackerUnicastAddressInfo::mAddressOrigin

The address origin (`OT_ADDRESS_ORIGIN_*` constants).

Definition at line `108` of file `include/openthread/history_tracker.h`

mEvent

```
otHistoryTrackerAddressEvent otHistoryTrackerUnicastAddressInfo::mEvent
```

Indicates the event (address is added/removed).

Definition at line `109` of file `include/openthread/history_tracker.h`

mScope

```
uint8_t otHistoryTrackerUnicastAddressInfo::mScope
```

The IPv6 scope.

Definition at line `110` of file `include/openthread/history_tracker.h`

mPreferred

```
bool otHistoryTrackerUnicastAddressInfo::mPreferred
```

If the address is preferred.

Definition at line `111` of file `include/openthread/history_tracker.h`

mValid

```
bool otHistoryTrackerUnicastAddressInfo::mValid
```

If the address is valid.

Definition at line `112` of file `include/openthread/history_tracker.h`

mRloc

```
bool otHistoryTrackerUnicastAddressInfo::mRloc
```

If the address is an RLOC.

Definition at line `113` of file `include/openthread/history_tracker.h`

otHistoryTrackerMulticastAddressInfo

Represent an IPv6 multicast address info.

Public Attributes

otIp6Address	mAddress The IPv6 multicast address.
uint8_t	mAddressOrigin The address origin (OT_ADDRESS_ORIGIN_* constants).
otHistoryTrackerAddressEvent	mEvent Indicates the event (address is added/removed).

Public Attribute Documentation

mAddress

```
otIp6Address otHistoryTrackerMulticastAddressInfo::mAddress
```

The IPv6 multicast address.

Definition at line 122 of file include/openthread/history_tracker.h

mAddressOrigin

```
uint8_t otHistoryTrackerMulticastAddressInfo::mAddressOrigin
```

The address origin (OT_ADDRESS_ORIGIN_* constants).

Definition at line 123 of file include/openthread/history_tracker.h

mEvent

```
otHistoryTrackerAddressEvent otHistoryTrackerMulticastAddressInfo::mEvent
```

Indicates the event (address is added/removed).

Definition at line 124 of file include/openthread/history_tracker.h

otHistoryTrackerMessageInfo

Represents a RX/TX IPv6 message info.

Some of the fields in this struct are applicable to a RX message or a TX message only, e.g., `mAveRxRss` is the average RSS of all fragment frames that form a received message and is only applicable for a RX message.

Public Attributes

uint16_t	mPayloadLength IPv6 payload length (exclude IP6 header itself).
uint16_t	mNeighborRloc16 RLOC16 of neighbor which sent/received the msg (`0xfffe` if no RLOC16).
otSockAddr	mSource Source IPv6 address and port (if UDP/TCP)
otSockAddr	mDestination Destination IPv6 address and port (if UDP/TCP).
uint16_t	mChecksum Message checksum (valid only for UDP/TCP/ICMP6).
uint8_t	mIpProto IP Protocol number (`OT_IP6_PROTO_*` enumeration).
uint8_t	mIcmp6Type ICMP6 type if msg is ICMP6, zero otherwise (`OT_ICMP6_TYPE_*` enumeration).
int8_t	mAveRxRss RSS of received message or OT_RADIO_INVALID_RSSI if not known.
bool	mLinkSecurity Indicates whether msg used link security.
bool	mTxSuccess Indicates TX success (e.g., ack received). Applicable for TX msg only.
uint8_t	mPriority Message priority (`OT_HISTORY_TRACKER_MSG_PRIORITY_*` enumeration).
bool	mRadioIeee802154 Indicates whether msg was sent/received over a 15.4 radio link.
bool	mRadioTrelUdp6 Indicates whether msg was sent/received over a TREL radio link.

Public Attribute Documentation

mPayloadLength

> uint16_t otHistoryTrackerMessageInfo::mPayloadLength

IPv6 payload length (exclude IP6 header itself).

Definition at line 148 of file include/openthread/history_tracker.h

mNeighborRloc16

```
uint16_t otHistoryTrackerMessageInfo::mNeighborRloc16
```

RLOC16 of neighbor which sent/received the msg (0xfffe if no RLOC16).

Definition at line 149 of file include/openthread/history_tracker.h

mSource

```
otSockAddr otHistoryTrackerMessageInfo::mSource
```

Source IPv6 address and port (if UDP/TCP)

Definition at line 150 of file include/openthread/history_tracker.h

mDestination

```
otSockAddr otHistoryTrackerMessageInfo::mDestination
```

Destination IPv6 address and port (if UDP/TCP).

Definition at line 151 of file include/openthread/history_tracker.h

mChecksum

```
uint16_t otHistoryTrackerMessageInfo::mChecksum
```

Message checksum (valid only for UDP/TCP/ICMP6).

Definition at line 152 of file include/openthread/history_tracker.h

mIpProto

```
uint8_t otHistoryTrackerMessageInfo::mIpProto
```

IP Protocol number (OT_IP6_PROTO_* enumeration).

Definition at line 153 of file include/openthread/history_tracker.h

mIcmp6Type

```
uint8_t otHistoryTrackerMessageInfo::mIcmp6Type
```

ICMP6 type if msg is ICMP6, zero otherwise (OT_ICMP6_TYPE_* enumeration).

Definition at line 154 of file include/openthread/history_tracker.h

mAveRxRss

```
int8_t otHistoryTrackerMessageInfo::mAveRxRss
```

RSS of received message or OT_RADIO_INVALID_RSSI if not known.

Definition at line 155 of file include/openthread/history_tracker.h

mLinkSecurity

```
bool otHistoryTrackerMessageInfo::mLinkSecurity
```

Indicates whether msg used link security.

Definition at line 156 of file include/openthread/history_tracker.h

mTxSuccess

```
bool otHistoryTrackerMessageInfo::mTxSuccess
```

Indicates TX success (e.g., ack received). Applicable for TX msg only.

Definition at line 157 of file include/openthread/history_tracker.h

mPriority

```
uint8_t otHistoryTrackerMessageInfo::mPriority
```

Message priority (OT_HISTORY_TRACKER_MSG_PRIORITY_* enumeration).

Definition at line 158 of file include/openthread/history_tracker.h

mRadioIeee802154

```
bool otHistoryTrackerMessageInfo::mRadioIeee802154
```

Indicates whether msg was sent/received over a 15.4 radio link.

Definition at line 159 of file include/openthread/history_tracker.h

mRadioTrelUdp6

```
bool otHistoryTrackerMessageInfo::mRadioTrelUdp6
```

Indicates whether msg was sent/received over a TREL radio link.

Definition at line 160 of file include/openthread/history_tracker.h

otHistoryTrackerNeighborInfo

Represents a neighbor info.

Public Attributes

otExtAddress	mExtAddress Neighbor's Extended Address.
uint16_t	mRloc16 Neighbor's RLOC16.
int8_t	mAverageRssi Average RSSI of rx frames from neighbor at the time of recording entry.
uint8_t	mEvent Indicates the event (`OT_HISTORY_TRACKER_NEIGHBOR_EVENT_*` enumeration).
bool	mRxOnWhenIdle Rx-on-when-idle.
bool	mFullThreadDevice Full Thread Device.
bool	mFullNetworkData Full Network Data.
bool	mIsChild Indicates whether or not the neighbor is a child.

Public Attribute Documentation

mExtAddress

```
otExtAddress otHistoryTrackerNeighborInfo::mExtAddress
```

Neighbor's Extended Address.

Definition at line `185` of file `include/openthread/history_tracker.h`

mRloc16

```
uint16_t otHistoryTrackerNeighborInfo::mRloc16
```

Neighbor's RLOC16.

Definition at line `186` of file `include/openthread/history_tracker.h`

mAverageRssi

```
int8_t otHistoryTrackerNeighborInfo::mAverageRssi
```

Average RSSI of rx frames from neighbor at the time of recording entry.

Definition at line 187 of file include/openthread/history_tracker.h

mEvent

```
uint8_t otHistoryTrackerNeighborInfo::mEvent
```

Indicates the event (OT_HISTORY_TRACKER_NEIGHBOR_EVENT_* enumeration).

Definition at line 188 of file include/openthread/history_tracker.h

mRxOnWhenIdle

```
bool otHistoryTrackerNeighborInfo::mRxOnWhenIdle
```

Rx-on-when-idle.

Definition at line 189 of file include/openthread/history_tracker.h

mFullThreadDevice

```
bool otHistoryTrackerNeighborInfo::mFullThreadDevice
```

Full Thread Device.

Definition at line 190 of file include/openthread/history_tracker.h

mFullNetworkData

```
bool otHistoryTrackerNeighborInfo::mFullNetworkData
```

Full Network Data.

Definition at line 191 of file include/openthread/history_tracker.h

mIsChild

```
bool otHistoryTrackerNeighborInfo::mIsChild
```

Indicates whether or not the neighbor is a child.

Definition at line 192 of file include/openthread/history_tracker.h

otHistoryTrackerRouterInfo

Represents a router table entry event.

Public Attributes

uint8_t	mEvent Router entry event (OT_HISTORY_TRACKER_ROUTER_EVENT_* enumeration).
uint8_t	mRouterId Router ID.
uint8_t	mNextHop Next Hop Router ID - OT_HISTORY_TRACKER_NO_NEXT_HOP if no next hop.
uint8_t	mOldPathCost Old path cost - OT_HISTORY_TRACKER_INFINITE_PATH_COST if infinite or unknown.
uint8_t	mPathCost New path cost - OT_HISTORY_TRACKER_INFINITE_PATH_COST if infinite or unknown.

Public Attribute Documentation

mEvent

```
uint8_t otHistoryTrackerRouterInfo::mEvent
```

Router entry event (OT_HISTORY_TRACKER_ROUTER_EVENT_* enumeration).

Definition at line 217 of file include/openthread/history_tracker.h

mRouterId

```
uint8_t otHistoryTrackerRouterInfo::mRouterId
```

Router ID.

Definition at line 218 of file include/openthread/history_tracker.h

mNextHop

```
uint8_t otHistoryTrackerRouterInfo::mNextHop
```

Next Hop Router ID - OT_HISTORY_TRACKER_NO_NEXT_HOP if no next hop.

Definition at line 219 of file include/openthread/history_tracker.h

mOldPathCost

```
uint8_t otHistoryTrackerRouterInfo::mOldPathCost
```

Old path cost - `OT_HISTORY_TRACKER_INFINITE_PATH_COST` if infinite or unknown.

Definition at line `220` of file `include/openthread/history_tracker.h`

mPathCost

```
uint8_t otHistoryTrackerRouterInfo::mPathCost
```

New path cost - `OT_HISTORY_TRACKER_INFINITE_PATH_COST` if infinite or unknown.

Definition at line `221` of file `include/openthread/history_tracker.h`

otHistoryTrackerOnMeshPrefixInfo

Represent a Network Data on mesh prefix info.

Public Attributes

otBorderRouterConfig	mPrefix The on mesh prefix entry.
otHistoryTrackerNetDataEvent	mEvent Indicates the event (added/removed).

Public Attribute Documentation

mPrefix

otBorderRouterConfig otHistoryTrackerOnMeshPrefixInfo::mPrefix

The on mesh prefix entry.

Definition at line 240 of file include/openthread/history_tracker.h

mEvent

otHistoryTrackerNetDataEvent otHistoryTrackerOnMeshPrefixInfo::mEvent

Indicates the event (added/removed).

Definition at line 241 of file include/openthread/history_tracker.h

otHistoryTrackerExternalRouteInfo

Represent a Network Data extern route info.

Public Attributes

otExternalRouteConfig	mRoute The external route entry.
otHistoryTrackerNetDataEvent	mEvent Indicates the event (added/removed).

Public Attribute Documentation

mRoute

```
otExternalRouteConfig otHistoryTrackerExternalRouteInfo::mRoute
```

The external route entry.

Definition at line 250 of file include/openthread/history_tracker.h

mEvent

```
otHistoryTrackerNetDataEvent otHistoryTrackerExternalRouteInfo::mEvent
```

Indicates the event (added/removed).

Definition at line 251 of file include/openthread/history_tracker.h

Jam Detection

Jam Detection

This module includes functions for signal jamming detection feature.

The functions in this module are available when jam-detection feature (`OPENTHREAD_CONFIG_JAM_DETECTION_ENABLE`) is enabled.

Typedefs

typedef void(* otJamDetectionCallback)(bool aJamState, void *aContext)
Pointer is called if signal jam detection is enabled and a jam is detected.

Functions

otError otJamDetectionSetRssiThreshold(otInstance *aInstance, int8_t aRssiThreshold)
Set the Jam Detection RSSI Threshold (in dBm).

int8_t otJamDetectionGetRssiThreshold(otInstance *aInstance)
Get the Jam Detection RSSI Threshold (in dBm).

otError otJamDetectionSetWindow(otInstance *aInstance, uint8_t aWindow)
Set the Jam Detection Detection Window (in seconds).

uint8_t otJamDetectionGetWindow(otInstance *aInstance)
Get the Jam Detection Detection Window (in seconds).

otError otJamDetectionSetBusyPeriod(otInstance *aInstance, uint8_t aBusyPeriod)
Set the Jam Detection Busy Period (in seconds).

uint8_t otJamDetectionGetBusyPeriod(otInstance *aInstance)
Get the Jam Detection Busy Period (in seconds)

otError otJamDetectionStart(otInstance *aInstance, otJamDetectionCallback aCallback, void *aContext)
Start the jamming detection.

otError otJamDetectionStop(otInstance *aInstance)
Stop the jamming detection.

bool otJamDetectionIsEnabled(otInstance *aInstance)
Get the Jam Detection Status (enabled/disabled)

bool otJamDetectionGetState(otInstance *aInstance)
Get the Jam Detection State.

uint64_t otJamDetectionGetHistoryBitmap(otInstance *aInstance)
Get the current history bitmap.

Typedef Documentation

otJamDetectionCallback

```
typedef void(* otJamDetectionCallback) (bool aJamState, void *aContext) )(bool aJamState, void *aContext)
```

Pointer is called if signal jam detection is enabled and a jam is detected.

Parameters

[in]	aJamState	Current jam state (`true` if jam is detected, `false` otherwise).
[in]	aContext	A pointer to application-specific context.

Definition at line `64` of file `include/openthread/jam_detection.h`

Function Documentation

otJamDetectionSetRssiThreshold

```
otError otJamDetectionSetRssiThreshold (otInstance *aInstance, int8_t aRssiThreshold)
```

Set the Jam Detection RSSI Threshold (in dBm).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aRssiThreshold	The RSSI threshold.

Definition at line `75` of file `include/openthread/jam_detection.h`

otJamDetectionGetRssiThreshold

```
int8_t otJamDetectionGetRssiThreshold (otInstance *aInstance)
```

Get the Jam Detection RSSI Threshold (in dBm).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Jam Detection RSSI Threshold.

Definition at line `84` of file `include/openthread/jam_detection.h`

otJamDetectionSetWindow

```
otError otJamDetectionSetWindow (otInstance *aInstance, uint8_t aWindow)
```

Set the Jam Detection Detection Window (in seconds).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aWindow	The Jam Detection window (valid range is 1 to 63)

Definition at line 96 of file include/openthread/jam_detection.h

otJamDetectionGetWindow

```
uint8_t otJamDetectionGetWindow (otInstance *aInstance)
```

Get the Jam Detection Detection Window (in seconds).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Jam Detection Window.

Definition at line 106 of file include/openthread/jam_detection.h

otJamDetectionSetBusyPeriod

```
otError otJamDetectionSetBusyPeriod (otInstance *aInstance, uint8_t aBusyPeriod)
```

Set the Jam Detection Busy Period (in seconds).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aBusyPeriod	The Jam Detection busy period (should be non-zero and less than or equal to Jam Detection Window)

The number of aggregate seconds within the detection window where the RSSI must be above threshold to trigger detection.

Definition at line 122 of file include/openthread/jam_detection.h

otJamDetectionGetBusyPeriod

```
uint8_t otJamDetectionGetBusyPeriod (otInstance *aInstance)
```

Get the Jam Detection Busy Period (in seconds)

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Jam Detection Busy Period.

Definition at line 132 of file include/openthread/jam_detection.h

otJamDetectionStart

```
otError otJamDetectionStart (otInstance *aInstance, otJamDetectionCallback aCallback, void *aContext)
```

Start the jamming detection.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aCallback	A pointer to a function called to notify of jamming state change.
[in]	aContext	A pointer to application-specific context.

Definition at line `145` of file `include/openthread/jam_detection.h`

otJamDetectionStop

```
otError otJamDetectionStop (otInstance *aInstance)
```

Stop the jamming detection.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Definition at line `156` of file `include/openthread/jam_detection.h`

otJamDetectionIsEnabled

```
bool otJamDetectionIsEnabled (otInstance *aInstance)
```

Get the Jam Detection Status (enabled/disabled)

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Jam Detection status (true if enabled, false otherwise).

Definition at line `166` of file `include/openthread/jam_detection.h`

otJamDetectionGetState

```
bool otJamDetectionGetState (otInstance *aInstance)
```

Get the Jam Detection State.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The Jam Detection state (`true` jam is detected, 'false' otherwise).

Definition at line `176` of file `include/openthread/jam_detection.h`

otJamDetectionGetHistoryBitmap

```
uint64_t otJamDetectionGetHistoryBitmap (otInstance *aInstance)
```

Get the current history bitmap.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

This value provides information about current state of jamming detection module for monitoring/debugging purpose. It returns a 64-bit value where each bit corresponds to one second interval starting with bit 0 for the most recent interval and bit 63 for the oldest intervals (63 sec earlier). The bit is set to 1 if the jamming detection module observed/detected high signal level during the corresponding one second interval.

Returns

- The current history bitmap.

Definition at line `193` of file `include/openthread/jam_detection.h`

Logging - Thread Stack

This module includes OpenThread logging related definitions.

Modules

otLogHexDumpInfo

Functions

otLogLevel otLoggingGetLevel(void)
Returns the current log level.

otError otLoggingSetLevel(otLogLevel aLogLevel)
Sets the log level.

void otLogCritPlat(const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1
Emits a log message at critical log level.

void void otLogWarnPlat(const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1
Emits a log message at warning log level.

void void void otLogNotePlat(const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1
Emits a log message at note log level.

void void void void otLogInfoPlat(const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1
Emits a log message at info log level.

void void void void void otLogDebgPlat(const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1
Emits a log message at debug log level.

void void void void void void otDumpCritPlat(const char *aText, const void *aData, uint16_t aDataLength)
Generates a memory dump at critical log level.

void otDumpWarnPlat(const char *aText, const void *aData, uint16_t aDataLength)
Generates a memory dump at warning log level.

void otDumpNotePlat(const char *aText, const void *aData, uint16_t aDataLength)
Generates a memory dump at note log level.

void otDumpInfoPlat(const char *aText, const void *aData, uint16_t aDataLength)
Generates a memory dump at info log level.

void otDumpDebgPlat(const char *aText, const void *aData, uint16_t aDataLength)
Generates a memory dump at debug log level.

void otLogPlat(otLogLevel aLogLevel, const char *aPlatModuleName, const char *aFormat,...)
OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(3
Emits a log message at given log level using a platform module name.

void void otLogPlatArgs(otLogLevel aLogLevel, const char *aPlatModuleName, const char *aFormat, va_list aArgs)
Emits a log message at given log level using a platform module name.

void	otLogCli(otLogLevel aLogLevel, const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(2 Emits a log message at a given log level.
otError	otLogGenerateNextHexDumpLine(otLogHexDumpInfo *aInfo) Generates the next hex dump line.

Macros

#define	OT_LOG_HEX_DUMP_LINE_SIZE 73 Hex dump line string size.

Function Documentation

otLoggingGetLevel

> otLogLevel otLoggingGetLevel (void)

Returns the current log level.

Parameters

N/A		

If dynamic log level feature `OPENTHREAD_CONFIG_LOG_LEVEL_DYNAMIC_ENABLE` is enabled, this function returns the currently set dynamic log level. Otherwise, this function returns the build-time configured log level.

Returns

- The log level.

Definition at line `64` of file `include/openthread/logging.h`

otLoggingSetLevel

> otError otLoggingSetLevel (otLogLevel aLogLevel)

Sets the log level.

Parameters

[in]	aLogLevel	The log level.

Note

- This function requires `OPENTHREAD_CONFIG_LOG_LEVEL_DYNAMIC_ENABLE=1`.

Definition at line `77` of file `include/openthread/logging.h`

otLogCritPlat

> void otLogCritPlat (const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1

Emits a log message at critical log level.

Parameters

[in]	aFormat	The format string.

[in]		Arguments for the format specification.

Is intended for use by platform. If `OPENTHREAD_CONFIG_LOG_PLATFORM` is not set or the current log level is below critical, this function does not emit any log message.

Definition at line `89` of file `include/openthread/logging.h`

otLogWarnPlat

```
void void otLogWarnPlat (const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1
```

Emits a log message at warning log level.

Parameters

[in]	aFormat	The format string.
[in]		Arguments for the format specification.

Is intended for use by platform. If `OPENTHREAD_CONFIG_LOG_PLATFORM` is not set or the current log level is below warning, this function does not emit any log message.

Definition at line `101` of file `include/openthread/logging.h`

otLogNotePlat

```
void void void otLogNotePlat (const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1
```

Emits a log message at note log level.

Parameters

[in]	aFormat	The format string.
[in]		Arguments for the format specification.

Is intended for use by platform. If `OPENTHREAD_CONFIG_LOG_PLATFORM` is not set or the current log level is below note, this function does not emit any log message.

Definition at line `113` of file `include/openthread/logging.h`

otLogInfoPlat

```
void void void void otLogInfoPlat (const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1
```

Emits a log message at info log level.

Parameters

[in]	aFormat	The format string.
[in]		Arguments for the format specification.

Is intended for use by platform. If `OPENTHREAD_CONFIG_LOG_PLATFORM` is not set or the current log level is below info, this function does not emit any log message.

Definition at line `125` of file `include/openthread/logging.h`

otLogDebgPlat

> void void void void void otLogDebgPlat (const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(1

Emits a log message at debug log level.

Parameters

[in]	aFormat	The format string.
[in]		Arguments for the format specification.

Is intended for use by platform. If `OPENTHREAD_CONFIG_LOG_PLATFORM` is not set or the current log level is below debug, this function does not emit any log message.

Definition at line `137` of file `include/openthread/logging.h`

otDumpCritPlat

> void void void void void void otDumpCritPlat (const char *aText, const void *aData, uint16_t aDataLength)

Generates a memory dump at critical log level.

Parameters

[in]	aText	A string that is printed before the bytes.
[in]	aData	A pointer to the data buffer.
[in]	aDataLength	Number of bytes in `aData`.

If `OPENTHREAD_CONFIG_LOG_PLATFORM` or `OPENTHREAD_CONFIG_LOG_PKT_DUMP` is not set or the current log level is below critical this function does not emit any log message.

Definition at line `150` of file `include/openthread/logging.h`

otDumpWarnPlat

> void otDumpWarnPlat (const char *aText, const void *aData, uint16_t aDataLength)

Generates a memory dump at warning log level.

Parameters

[in]	aText	A string that is printed before the bytes.
[in]	aData	A pointer to the data buffer.
[in]	aDataLength	Number of bytes in `aData`.

If `OPENTHREAD_CONFIG_LOG_PLATFORM` or `OPENTHREAD_CONFIG_LOG_PKT_DUMP` is not set or the current log level is below warning this function does not emit any log message.

Definition at line `163` of file `include/openthread/logging.h`

otDumpNotePlat

> void otDumpNotePlat (const char *aText, const void *aData, uint16_t aDataLength)

Generates a memory dump at note log level.

Parameters

[in]	aText	A string that is printed before the bytes.
[in]	aData	A pointer to the data buffer.
[in]	aDataLength	Number of bytes in `aData`.

If `OPENTHREAD_CONFIG_LOG_PLATFORM` or `OPENTHREAD_CONFIG_LOG_PKT_DUMP` is not set or the current log level is below note this function does not emit any log message.

Definition at line `176` of file `include/openthread/logging.h`

otDumpInfoPlat

```
void otDumpInfoPlat (const char *aText, const void *aData, uint16_t aDataLength)
```

Generates a memory dump at info log level.

Parameters

[in]	aText	A string that is printed before the bytes.
[in]	aData	A pointer to the data buffer.
[in]	aDataLength	Number of bytes in `aData`.

If `OPENTHREAD_CONFIG_LOG_PLATFORM` or `OPENTHREAD_CONFIG_LOG_PKT_DUMP` is not set or the current log level is below info this function does not emit any log message.

Definition at line `189` of file `include/openthread/logging.h`

otDumpDebgPlat

```
void otDumpDebgPlat (const char *aText, const void *aData, uint16_t aDataLength)
```

Generates a memory dump at debug log level.

Parameters

[in]	aText	A string that is printed before the bytes.
[in]	aData	A pointer to the data buffer.
[in]	aDataLength	Number of bytes in `aData`.

If `OPENTHREAD_CONFIG_LOG_PLATFORM` or `OPENTHREAD_CONFIG_LOG_PKT_DUMP` is not set or the current log level is below debug this function does not emit any log message.

Definition at line `202` of file `include/openthread/logging.h`

otLogPlat

```
void otLogPlat (otLogLevel aLogLevel, const char *aPlatModuleName, const char *aFormat,...)
OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(3
```

Emits a log message at given log level using a platform module name.

Parameters

[in]	aLogLevel	The log level.
[in]	aPlatModuleName	The platform sub-module name.
[in]	aFormat	The format string.
[in]		Arguments for the format specification.

This is is intended for use by platform. If `OPENTHREAD_CONFIG_LOG_PLATFORM` is not set or the current log level is below `aLogLevel` , this function does not emit any log message.

The `aPlatModuleName` name is used to determine the log module name in the emitted log message, following the `P-{PlatModuleName}---` format. This means that the prefix string "P-" is added to indicate that this is a platform sub-module, followed by the next 12 characters of the `PlatModuleName` string, with padded hyphens `-` at the end to ensure that the region name is 14 characters long.

Definition at line `221` of file `include/openthread/logging.h`

otLogPlatArgs

> void void otLogPlatArgs (otLogLevel aLogLevel, const char *aPlatModuleName, const char *aFormat, va_list aArgs)

Emits a log message at given log level using a platform module name.

Parameters

[in]	aLogLevel	The log level.
[in]	aPlatModuleName	The platform sub-module name.
[in]	aFormat	The format string.
[in]	aArgs	Arguments for the format specification.

This is is intended for use by platform. If `OPENTHREAD_CONFIG_LOG_PLATFORM` is not set or the current log level is below `aLogLevel` , this function does not emit any log message.

The `aPlatModuleName` name is used to determine the log module name in the emitted log message, following the `P-{PlatModuleName}---` format. This means that the prefix string "P-" is added to indicate that this is a platform sub-module, followed by the next 12 characters of the `PlatModuleName` string, with padded hyphens `-` at the end to ensure that the region name is 14 characters long.

Definition at line `241` of file `include/openthread/logging.h`

otLogCli

> void otLogCli (otLogLevel aLogLevel, const char *aFormat,...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(2

Emits a log message at a given log level.

Parameters

[in]	aLogLevel	The log level.
[in]	aFormat	The format string.
[in]		Arguments for the format specification.

Is intended for use by CLI only. If `OPENTHREAD_CONFIG_LOG_CLI` is not set or the current log level is below the given log level, this function does not emit any log message.

Definition at line 254 of file include/openthread/logging.h

otLogGenerateNextHexDumpLine

```
otError otLogGenerateNextHexDumpLine (otLogHexDumpInfo *aInfo)
```

Generates the next hex dump line.

Parameters

[inout]	aInfo	A pointer to otLogHexDumpInfo to use to generate hex dump.

Can call this method back-to-back to generate the hex dump output line by line. On the first call the mIterator field in aInfo MUST be set to zero.

Here is an example of the generated hex dump output:

"=========================[{mTitle} len=070]============================" "| 41 D8 87 34 12 FF FF 25 | 4C 57 DA F2 FB 2F 62 7F | A..4...%LW.../b. |" "| 3B 01 F0 4D 4C 4D 4C 54 | 4F 00 15 15 00 00 00 00 | ;..MLMLTO..... |" "| 00 00 00 01 80 DB 60 82 | 7E 33 72 3B CC B3 A1 84 | ......`.~3r;.... |" "| 3B E6 AD B2 0B 45 E7 45 | C5 B9 00 1A CB 2D 6D 1C | ;....E.E.....-m. |" "| 10 3E 3C F5 D3 70 | | .><..p |" "--"

Definition at line 293 of file include/openthread/logging.h

Macro Definition Documentation

OT_LOG_HEX_DUMP_LINE_SIZE

```
#define OT_LOG_HEX_DUMP_LINE_SIZE
```

Value:

```
73
```

Hex dump line string size.

Definition at line 256 of file include/openthread/logging.h

otLogHexDumpInfo

Represents information used for generating hex dump output.

Public Attributes

const uint8_t *	mDataBytes The data byes.
uint16_t	mDataLength The data length (number of bytes in `mDataBytes`)
const char *	mTitle Title string to add table header (MUST NOT be `NULL`).
char	mLine Buffer to output one line of generated hex dump.
uint16_t	mIterator Iterator used by OT stack. MUST be initialized to zero.

Public Attribute Documentation

mDataBytes

```
const uint8_t* otLogHexDumpInfo::mDataBytes
```

The data byes.

Definition at line `264` of file `include/openthread/logging.h`

mDataLength

```
uint16_t otLogHexDumpInfo::mDataLength
```

The data length (number of bytes in `mDataBytes`)

Definition at line `265` of file `include/openthread/logging.h`

mTitle

```
const char* otLogHexDumpInfo::mTitle
```

Title string to add table header (MUST NOT be `NULL`).

Definition at line `266` of file `include/openthread/logging.h`

mLine

```
char otLogHexDumpInfo::mLine[OT_LOG_HEX_DUMP_LINE_SIZE]
```

Buffer to output one line of generated hex dump.

Definition at line `267` of file `include/openthread/logging.h`

mIterator

```
uint16_t otLogHexDumpInfo::mIterator
```

Iterator used by OT stack. MUST be initialized to zero.

Definition at line `268` of file `include/openthread/logging.h`

Mesh Diagnostics

Mesh Diagnostics

This module includes definitions and functions for Mesh Diagnostics.

The Mesh Diagnostics APIs require `OPENTHREAD_CONFIG_MESH_DIAG_ENABLE` and `OPENTHREAD_FTD`.

Modules

otMeshDiagDiscoverConfig

otMeshDiagRouterInfo

otMeshDiagChildInfo

otMeshDiagChildEntry

otMeshDiagRouterNeighborEntry

Typedefs

typedef struct otMeshDiagDiscoverConfig — otMeshDiagDiscoverConfig
Represents the set of configurations used when discovering mesh topology indicating which items to discover.

typedef struct otMeshDiagIp6AddrIterator — otMeshDiagIp6AddrIterator
An opaque iterator to iterate over list of IPv6 addresses of a router.

typedef struct otMeshDiagChildIterator — otMeshDiagChildIterator
An opaque iterator to iterate over list of children of a router.

typedef struct otMeshDiagRouterInfo — otMeshDiagRouterInfo
Represents information about a router in Thread mesh discovered using `otMeshDiagDiscoverTopology()`.

typedef struct otMeshDiagChildInfo — otMeshDiagChildInfo
Represents information about a discovered child in Thread mesh using `otMeshDiagDiscoverTopology()`.

typedef void(* otMeshDiagDiscoverCallback)(otError aError, otMeshDiagRouterInfo *aRouterInfo, void *aContext)
Pointer type represents the callback used by `otMeshDiagDiscoverTopology()` to provide information about a discovered router.

typedef struct otMeshDiagChildEntry — otMeshDiagChildEntry
Represents information about a child entry from `otMeshDiagQueryChildTable()`.

typedef void(* otMeshDiagQueryChildTableCallback)(otError aError, const otMeshDiagChildEntry *aChildEntry, void *aContext)
Represents the callback used by `otMeshDiagQueryChildTable()` to provide information about child table entries.

typedef void(* otMeshDiagChildIp6AddrsCallback)(otError aError, uint16_t aChildRloc16, otMeshDiagIp6AddrIterator *aIp6AddrIterator, void *aContext)
Represents the callback used by `otMeshDiagQueryChildrenIp6Addrs()` to provide information about an MTD child and its list of IPv6 addresses.

typedef struct otMeshDiagRouterNeighborEntry otMeshDiagRouterNeighborEntry
Represents information about a router neighbor entry from `otMeshDiagQueryRouterNeighborTable()` .

typedef void(* otMeshDiagQueryRouterNeighborTableCallback)(otError aError, const otMeshDiagRouterNeighborEntry *aNeighborEntry, void *aContext)
Represents the callback used by `otMeshDiagQueryRouterNeighborTable()` to provide information about neighbor router table entries.

Functions

otError otMeshDiagDiscoverTopology(otInstance *aInstance, const otMeshDiagDiscoverConfig *aConfig, otMeshDiagDiscoverCallback aCallback, void *aContext)
Starts network topology discovery.

void otMeshDiagCancel(otInstance *aInstance)
Cancels an ongoing topology discovery if there is one, otherwise no action.

otError otMeshDiagGetNextIp6Address(otMeshDiagIp6AddrIterator *aIterator, otIp6Address *aIp6Address)
Iterates through the discovered IPv6 addresses of a router or an MTD child.

otError otMeshDiagGetNextChildInfo(otMeshDiagChildIterator *aIterator, otMeshDiagChildInfo *aChildInfo)
Iterates through the discovered children of a router.

otError otMeshDiagQueryChildTable(otInstance *aInstance, uint16_t aRloc16, otMeshDiagQueryChildTableCallback aCallback, void *aContext)
Starts query for child table for a given router.

otError otMeshDiagQueryChildrenIp6Addrs(otInstance *aInstance, uint16_t aRloc16, otMeshDiagChildIp6AddrsCallback aCallback, void *aContext)
Sends a query to a parent to retrieve the IPv6 addresses of all its MTD children.

otError otMeshDiagQueryRouterNeighborTable(otInstance *aInstance, uint16_t aRloc16, otMeshDiagQueryRouterNeighborTableCallback aCallback, void *aContext)
Starts query for router neighbor table for a given router.

Macros

#define OT_MESH_DIAG_VERSION_UNKNOWN 0xffff
Specifies that Thread Version is unknown.

Typedef Documentation

otMeshDiagDiscoverConfig

```
typedef struct otMeshDiagDiscoverConfig otMeshDiagDiscoverConfig
```

Represents the set of configurations used when discovering mesh topology indicating which items to discover.

Definition at line `66` of file `include/openthread/mesh_diag.h`

otMeshDiagIp6AddrIterator

```
typedef struct otMeshDiagIp6AddrIterator otMeshDiagIp6AddrIterator
```

An opaque iterator to iterate over list of IPv6 addresses of a router.

Pointers to instance of this type are provided in `otMeshDiagRouterInfo`.

Definition at line `74` of file `include/openthread/mesh_diag.h`

otMeshDiagChildIterator

```
typedef struct otMeshDiagChildIterator otMeshDiagChildIterator
```

An opaque iterator to iterate over list of children of a router.

Pointers to instance of this type are provided in `otMeshDiagRouterInfo`.

Definition at line `82` of file `include/openthread/mesh_diag.h`

otMeshDiagRouterInfo

```
typedef struct otMeshDiagRouterInfo otMeshDiagRouterInfo
```

Represents information about a router in Thread mesh discovered using `otMeshDiagDiscoverTopology()`.

Definition at line `142` of file `include/openthread/mesh_diag.h`

otMeshDiagChildInfo

```
typedef struct otMeshDiagChildInfo otMeshDiagChildInfo
```

Represents information about a discovered child in Thread mesh using `otMeshDiagDiscoverTopology()`.

Definition at line `155` of file `include/openthread/mesh_diag.h`

otMeshDiagDiscoverCallback

```
typedef void(* otMeshDiagDiscoverCallback) (otError aError, otMeshDiagRouterInfo *aRouterInfo, void *aContext) )
(otError aError, otMeshDiagRouterInfo *aRouterInfo, void *aContext)
```

Pointer type represents the callback used by `otMeshDiagDiscoverTopology()` to provide information about a discovered router.

Parameters

[in]	aError	OT_ERROR_PENDING Indicates there are more routers to be discovered. OT_ERROR_NONE Indicates this is the last router and mesh discovery is done. OT_ERROR_RESPONSE_TIMEOUT Timed out waiting for response from one or more routers.
[in]	aRouterInfo	The discovered router info (can be null if `aError` is OT_ERROR_RESPONSE_TIMEOUT).
[in]	aContext	Application-specific context.

When `aError` is `OT_ERROR_PENDING`, it indicates that the discovery is not yet finished and there will be more routers to discover and the callback will be invoked again.

Definition at line `171` of file `include/openthread/mesh_diag.h`

otMeshDiagChildEntry

```
typedef struct otMeshDiagChildEntry otMeshDiagChildEntry
```

Represents information about a child entry from `otMeshDiagQueryChildTable()`.

`mSupportsErrRate` indicates whether or not the error tracking feature is supported and `mFrameErrorRate` and `mMessageErrorRate` values are valid. The frame error rate tracks frame tx errors (towards the child) at MAC layer, while `mMessageErrorRate` tracks the IPv6 message error rate (above MAC layer and after MAC retries) when an IPv6 message is dropped. For example, if the message is large and requires 6LoWPAN fragmentation, message tx is considered as failed if one of its fragment frame tx fails (for example, never acked).

Definition at line `265` of file `include/openthread/mesh_diag.h`

otMeshDiagQueryChildTableCallback

```
typedef void(* otMeshDiagQueryChildTableCallback) (otError aError, const otMeshDiagChildEntry *aChildEntry, void *aContext) )(otError aError, const otMeshDiagChildEntry *aChildEntry, void *aContext)
```

Represents the callback used by `otMeshDiagQueryChildTable()` to provide information about child table entries.

Parameters

[in]	aError	OT_ERROR_PENDING Indicates there are more entries in the table. OT_ERROR_NONE Indicates the table is finished. OT_ERROR_RESPONSE_TIMEOUT Timed out waiting for response.
[in]	aChildEntry	The child entry (can be null if `aError` is OT_ERROR_RESPONSE_TIMEOUT or OT_ERROR_NONE).
[in]	aContext	Application-specific context.

When `aError` is `OT_ERROR_PENDING`, it indicates that the table still has more entries and the callback will be invoked again.

Definition at line `280` of file `include/openthread/mesh_diag.h`

otMeshDiagChildIp6AddrsCallback

```
typedef void(* otMeshDiagChildIp6AddrsCallback) (otError aError, uint16_t aChildRloc16, otMeshDiagIp6AddrIterator *aIp6AddrIterator, void *aContext) )(otError aError, uint16_t aChildRloc16, otMeshDiagIp6AddrIterator *aIp6AddrIterator, void *aContext)
```

Represents the callback used by `otMeshDiagQueryChildrenIp6Addrs()` to provide information about an MTD child and its list of IPv6 addresses.

Parameters

[in]	aError	OT_ERROR_PENDING Indicates there are more children in the table. OT_ERROR_NONE Indicates the table is finished. OT_ERROR_RESPONSE_TIMEOUT Timed out waiting for response.
[in]	aChildRloc16	The RLOC16 of the child. `0xfffe` is used on `OT_ERROR_RESPONSE_TIMEOUT`.
[in]	aIp6AddrIterator	An iterator to go through the IPv6 addresses of the child with `aRloc` using `otMeshDiagGetNextIp6Address()`. Set to NULL on `OT_ERROR_RESPONSE_TIMEOUT`.

[in]	aContext	Application-specific context.

When `aError` is `OT_ERROR_PENDING`, it indicates that there are more children and the callback will be invoked again.

Definition at line `320` of file `include/openthread/mesh_diag.h`

otMeshDiagRouterNeighborEntry

```
typedef struct otMeshDiagRouterNeighborEntry otMeshDiagRouterNeighborEntry
```

Represents information about a router neighbor entry from `otMeshDiagQueryRouterNeighborTable()`.

`mSupportsErrRate` indicates whether or not the error tracking feature is supported and `mFrameErrorRate` and `mMessageErrorRate` values are valid. The frame error rate tracks frame tx errors (towards the child) at MAC layer, while `mMessageErrorRate` tracks the IPv6 message error rate (above MAC layer and after MAC retries) when an IPv6 message is dropped. For example, if the message is large and requires 6LoWPAN fragmentation, message tx is considered as failed if one of its fragment frame tx fails (for example, never acked).

Definition at line `367` of file `include/openthread/mesh_diag.h`

otMeshDiagQueryRouterNeighborTableCallback

```
typedef void(* otMeshDiagQueryRouterNeighborTableCallback) (otError aError, const otMeshDiagRouterNeighborEntry *aNeighborEntry, void *aContext) )(otError aError, const otMeshDiagRouterNeighborEntry *aNeighborEntry, void *aContext)
```

Represents the callback used by `otMeshDiagQueryRouterNeighborTable()` to provide information about neighbor router table entries.

Parameters

[in]	aError	OT_ERROR_PENDING Indicates there are more entries in the table. OT_ERROR_NONE Indicates the table is finished. OT_ERROR_RESPONSE_TIMEOUT Timed out waiting for response.
[in]	aNeighborEntry	The neighbor entry (can be null if `aError` is RESPONSE_TIMEOUT or NONE).
[in]	aContext	Application-specific context.

When `aError` is `OT_ERROR_PENDING`, it indicates that the table still has more entries and the callback will be invoked again.

Definition at line `383` of file `include/openthread/mesh_diag.h`

Function Documentation

otMeshDiagDiscoverTopology

```
otError otMeshDiagDiscoverTopology (otInstance *aInstance, const otMeshDiagDiscoverConfig *aConfig, otMeshDiagDiscoverCallback aCallback, void *aContext)
```

Starts network topology discovery.

Parameters

[in]	aInstance	The OpenThread instance.
[in]	aConfig	The configuration to use for discovery (e.g., which items to discover).
[in]	aCallback	The callback to report the discovered routers.

[in]	aContext	A context to pass in `aCallback`.

Definition at line `187` of file `include/openthread/mesh_diag.h`

otMeshDiagCancel

```
void otMeshDiagCancel (otInstance *aInstance)
```

Cancels an ongoing topology discovery if there is one, otherwise no action.

Parameters

N/A	aInstance	

When ongoing discovery is cancelled, the callback from `otMeshDiagDiscoverTopology()` will not be called anymore.

Definition at line `198` of file `include/openthread/mesh_diag.h`

otMeshDiagGetNextIp6Address

```
otError otMeshDiagGetNextIp6Address (otMeshDiagIp6AddrIterator *aIterator, otIp6Address *aIp6Address)
```

Iterates through the discovered IPv6 addresses of a router or an MTD child.

Parameters

[inout]	aIterator	The address iterator to use.
[out]	aIp6Address	A pointer to return the next IPv6 address (if any).

MUST be used

- from the callback `otMeshDiagDiscoverCallback()` and use the `mIp6AddrIterator` from the `aRouterInfo` struct that is provided as input to the callback, or
- from the callback `otMeshDiagChildIp6AddrsCallback()` along with provided `aIp6AddrIterator`.

Definition at line `215` of file `include/openthread/mesh_diag.h`

otMeshDiagGetNextChildInfo

```
otError otMeshDiagGetNextChildInfo (otMeshDiagChildIterator *aIterator, otMeshDiagChildInfo *aChildInfo)
```

Iterates through the discovered children of a router.

Parameters

[inout]	aIterator	The address iterator to use.
[out]	aChildInfo	A pointer to return the child info (if any).

This function MUST be used from the callback `otMeshDiagDiscoverCallback()` and use the `mChildIterator` from the `aRouterInfo` struct that is provided as input to the callback.

Definition at line `230` of file `include/openthread/mesh_diag.h`

otMeshDiagQueryChildTable

> otError otMeshDiagQueryChildTable (otInstance *aInstance, uint16_t aRloc16, otMeshDiagQueryChildTableCallback aCallback, void *aContext)

Starts query for child table for a given router.

Parameters

[in]	aInstance	The OpenThread instance.
[in]	aRloc16	The RLOC16 of router to query.
[in]	aCallback	The callback to report the queried child table.
[in]	aContext	A context to pass in `aCallback`.

Definition at line `299` of file `include/openthread/mesh_diag.h`

otMeshDiagQueryChildrenIp6Addrs

> otError otMeshDiagQueryChildrenIp6Addrs (otInstance *aInstance, uint16_t aRloc16, otMeshDiagChildIp6AddrsCallback aCallback, void *aContext)

Sends a query to a parent to retrieve the IPv6 addresses of all its MTD children.

Parameters

[in]	aInstance	The OpenThread instance.
[in]	aRloc16	The RLOC16 of parent to query.
[in]	aCallback	The callback to report the queried child IPv6 address list.
[in]	aContext	A context to pass in `aCallback`.

Definition at line `340` of file `include/openthread/mesh_diag.h`

otMeshDiagQueryRouterNeighborTable

> otError otMeshDiagQueryRouterNeighborTable (otInstance *aInstance, uint16_t aRloc16, otMeshDiagQueryRouterNeighborTableCallback aCallback, void *aContext)

Starts query for router neighbor table for a given router.

Parameters

[in]	aInstance	The OpenThread instance.
[in]	aRloc16	The RLOC16 of router to query.
[in]	aCallback	The callback to report the queried table.
[in]	aContext	A context to pass in `aCallback`.

Definition at line `402` of file `include/openthread/mesh_diag.h`

Macro Definition Documentation

OT_MESH_DIAG_VERSION_UNKNOWN

> #define OT_MESH_DIAG_VERSION_UNKNOWN

Value:

```
0xffff
```

Specifies that Thread Version is unknown.

This is used in `otMeshDiagRouterInfo` for `mVersion` property when device does not provide its version. This indicates that device is likely running 1.3.0 (version value 4) or earlier.

Definition at line `91` of file `include/openthread/mesh_diag.h`

otMeshDiagDiscoverConfig

Represents the set of configurations used when discovering mesh topology indicating which items to discover.

Public Attributes

bool mDiscoverIp6Addresses
Whether or not to discover IPv6 addresses of every router.

bool mDiscoverChildTable
Whether or not to discover children of every router.

Public Attribute Documentation

mDiscoverIp6Addresses

```
bool otMeshDiagDiscoverConfig::mDiscoverIp6Addresses
```

Whether or not to discover IPv6 addresses of every router.

Definition at line 64 of file include/openthread/mesh_diag.h

mDiscoverChildTable

```
bool otMeshDiagDiscoverConfig::mDiscoverChildTable
```

Whether or not to discover children of every router.

Definition at line 65 of file include/openthread/mesh_diag.h

otMeshDiagRouterInfo

Represents information about a router in Thread mesh discovered using `otMeshDiagDiscoverTopology()`.

Public Attributes

otExtAddress	mExtAddress Extended MAC address.
uint16_t	mRloc16 RLOC16.
uint8_t	mRouterId Router ID.
uint16_t	mVersion Thread Version. `OT_MESH_DIAG_VERSION_UNKNOWN` if unknown.
bool	mIsThisDevice Whether router is this device itself.
bool	mIsThisDeviceParent Whether router is parent of this device (when device is a child).
bool	mIsLeader Whether router is leader.
bool	mIsBorderRouter Whether router acts as a border router providing ext connectivity.
uint8_t	mLinkQualities Provides the link quality from this router to other routers, also indicating whether a link is established between the routers.
otMeshDiagIp6AddrIterator *	mIp6AddrIterator A pointer to an iterator to go through the list of IPv6 addresses of the router.
otMeshDiagChildIterator *	mChildIterator A pointer to an iterator to go through the list of children of the router.

Public Attribute Documentation

mExtAddress

```
otExtAddress otMeshDiagRouterInfo::mExtAddress
```

Extended MAC address.

Definition at line `99` of file `include/openthread/mesh_diag.h`

mRloc16

```
uint16_t otMeshDiagRouterInfo::mRloc16
```

RLOC16.

Definition at line 100 of file include/openthread/mesh_diag.h

mRouterId

```
uint8_t otMeshDiagRouterInfo::mRouterId
```

Router ID.

Definition at line 101 of file include/openthread/mesh_diag.h

mVersion

```
uint16_t otMeshDiagRouterInfo::mVersion
```

Thread Version. OT_MESH_DIAG_VERSION_UNKNOWN if unknown.

Definition at line 102 of file include/openthread/mesh_diag.h

mIsThisDevice

```
bool otMeshDiagRouterInfo::mIsThisDevice
```

Whether router is this device itself.

Definition at line 103 of file include/openthread/mesh_diag.h

mIsThisDeviceParent

```
bool otMeshDiagRouterInfo::mIsThisDeviceParent
```

Whether router is parent of this device (when device is a child).

Definition at line 104 of file include/openthread/mesh_diag.h

mIsLeader

```
bool otMeshDiagRouterInfo::mIsLeader
```

Whether router is leader.

Definition at line 105 of file include/openthread/mesh_diag.h

mIsBorderRouter

```
bool otMeshDiagRouterInfo::mIsBorderRouter
```

Whether router acts as a border router providing ext connectivity.

Definition at line 106 of file include/openthread/mesh_diag.h

mLinkQualities

```
uint8_t otMeshDiagRouterInfo::mLinkQualities[OT_NETWORK_MAX_ROUTER_ID+1]
```

Provides the link quality from this router to other routers, also indicating whether a link is established between the routers.

The array is indexed based on Router ID. `mLinkQualities[routerId]` indicates the incoming link quality, the router sees to the router with `routerId`. Link quality is a value in [0, 3]. Value zero indicates no link. Larger value indicate better link quality (as defined by Thread specification).

Definition at line 117 of file include/openthread/mesh_diag.h

mIp6AddrIterator

```
otMeshDiagIp6AddrIterator* otMeshDiagRouterInfo::mIp6AddrIterator
```

A pointer to an iterator to go through the list of IPv6 addresses of the router.

The pointer is valid only while `otMeshDiagRouterInfo` is valid. It can be used in `otMeshDiagGetNextIp6Address` to iterate through the IPv6 addresses.

The pointer can be NULL when there was no request to discover IPv6 addresses (in `otMeshDiagDiscoverConfig`) or if the router did not provide the list.

Definition at line 129 of file include/openthread/mesh_diag.h

mChildIterator

```
otMeshDiagChildIterator* otMeshDiagRouterInfo::mChildIterator
```

A pointer to an iterator to go through the list of children of the router.

The pointer is valid only while `otMeshDiagRouterInfo` is valid. It can be used in `otMeshDiagGetNextChildInfo` to iterate through the children of the router.

The pointer can be NULL when there was no request to discover children (in `otMeshDiagDiscoverConfig`) or if the router did not provide the list.

Definition at line 141 of file include/openthread/mesh_diag.h

otMeshDiagChildInfo

Represents information about a discovered child in Thread mesh using `otMeshDiagDiscoverTopology()`.

Public Attributes

uint16_t	mRloc16 RLOC16.
otLinkModeConfig	mMode Device mode.
uint8_t	mLinkQuality Incoming link quality to child from parent.
bool	mIsThisDevice Whether child is this device itself.
bool	mIsBorderRouter Whether child acts as a border router providing ext connectivity.

Public Attribute Documentation

mRloc16

```
uint16_t otMeshDiagChildInfo::mRloc16
```

RLOC16.

Definition at line `150` of file `include/openthread/mesh_diag.h`

mMode

```
otLinkModeConfig otMeshDiagChildInfo::mMode
```

Device mode.

Definition at line `151` of file `include/openthread/mesh_diag.h`

mLinkQuality

```
uint8_t otMeshDiagChildInfo::mLinkQuality
```

Incoming link quality to child from parent.

Definition at line `152` of file `include/openthread/mesh_diag.h`

mIsThisDevice

bool otMeshDiagChildInfo::mIsThisDevice

Whether child is this device itself.

Definition at line 153 of file include/openthread/mesh_diag.h

mIsBorderRouter

bool otMeshDiagChildInfo::mIsBorderRouter

Whether child acts as a border router providing ext connectivity.

Definition at line 154 of file include/openthread/mesh_diag.h

otMeshDiagChildEntry

Represents information about a child entry from `otMeshDiagQueryChildTable()`.

`mSupportsErrRate` indicates whether or not the error tracking feature is supported and `mFrameErrorRate` and `mMessageErrorRate` values are valid. The frame error rate tracks frame tx errors (towards the child) at MAC layer, while `mMessageErrorRate` tracks the IPv6 message error rate (above MAC layer and after MAC retries) when an IPv6 message is dropped. For example, if the message is large and requires 6LoWPAN fragmentation, message tx is considered as failed if one of its fragment frame tx fails (for example, never acked).

Public Attributes

bool	mRxOnWhenIdle Is rx-on when idle (vs sleepy).
bool	mDeviceTypeFtd Is device FTD (vs MTD).
bool	mFullNetData Whether device gets full Network Data (vs stable sub-set).
bool	mCslSynchronized Is CSL capable and CSL synchronized.
bool	mSupportsErrRate `mFrameErrorRate` and `mMessageErrorRate` values are valid.
uint16_t	mRloc16 RLOC16.
otExtAddress	mExtAddress Extended Address.
uint16_t	mVersion Version.
uint32_t	mTimeout Timeout in seconds.
uint32_t	mAge Seconds since last heard from the child.
uint32_t	mConnectionTime Seconds since child attach.
uint16_t	mSupervisionInterval Supervision interval in seconds. Zero to indicate not used.
uint8_t	mLinkMargin Link Margin in dB.
int8_t	mAverageRssi Average RSSI.
int8_t	mLastRssi RSSI of last received frame.

uint16_t	mFrameErrorRate Frame error rate (0x0000->0%, 0xffff->100%).
uint16_t	mMessageErrorRate (IPv6) msg error rate (0x0000->0%, 0xffff->100%).
uint16_t	mQueuedMessageCount Number of queued messages for indirect tx to child.
uint16_t	mCslPeriod CSL Period in unit of 10-symbols-time. Zero indicates CSL is disabled.
uint32_t	mCslTimeout CSL Timeout in seconds.
uint8_t	mCslChannel CSL channel.

Public Attribute Documentation

mRxOnWhenIdle

```
bool otMeshDiagChildEntry::mRxOnWhenIdle
```

Is rx-on when idle (vs sleepy).

Definition at line 244 of file include/openthread/mesh_diag.h

mDeviceTypeFtd

```
bool otMeshDiagChildEntry::mDeviceTypeFtd
```

Is device FTD (vs MTD).

Definition at line 245 of file include/openthread/mesh_diag.h

mFullNetData

```
bool otMeshDiagChildEntry::mFullNetData
```

Whether device gets full Network Data (vs stable sub-set).

Definition at line 246 of file include/openthread/mesh_diag.h

mCslSynchronized

```
bool otMeshDiagChildEntry::mCslSynchronized
```

Is CSL capable and CSL synchronized.

Definition at line 247 of file include/openthread/mesh_diag.h

mSupportsErrRate

```
bool otMeshDiagChildEntry::mSupportsErrRate
```

`mFrameErrorRate` and `mMessageErrorRate` values are valid.

Definition at line `248` of file `include/openthread/mesh_diag.h`

mRloc16

```
uint16_t otMeshDiagChildEntry::mRloc16
```

RLOC16.

Definition at line `249` of file `include/openthread/mesh_diag.h`

mExtAddress

```
otExtAddress otMeshDiagChildEntry::mExtAddress
```

Extended Address.

Definition at line `250` of file `include/openthread/mesh_diag.h`

mVersion

```
uint16_t otMeshDiagChildEntry::mVersion
```

Version.

Definition at line `251` of file `include/openthread/mesh_diag.h`

mTimeout

```
uint32_t otMeshDiagChildEntry::mTimeout
```

Timeout in seconds.

Definition at line `252` of file `include/openthread/mesh_diag.h`

mAge

```
uint32_t otMeshDiagChildEntry::mAge
```

Seconds since last heard from the child.

Definition at line `253` of file `include/openthread/mesh_diag.h`

mConnectionTime

uint32_t otMeshDiagChildEntry::mConnectionTime

Seconds since child attach.

Definition at line 254 of file include/openthread/mesh_diag.h

mSupervisionInterval

uint16_t otMeshDiagChildEntry::mSupervisionInterval

Supervision interval in seconds. Zero to indicate not used.

Definition at line 255 of file include/openthread/mesh_diag.h

mLinkMargin

uint8_t otMeshDiagChildEntry::mLinkMargin

Link Margin in dB.

Definition at line 256 of file include/openthread/mesh_diag.h

mAverageRssi

int8_t otMeshDiagChildEntry::mAverageRssi

Average RSSI.

Definition at line 257 of file include/openthread/mesh_diag.h

mLastRssi

int8_t otMeshDiagChildEntry::mLastRssi

RSSI of last received frame.

Definition at line 258 of file include/openthread/mesh_diag.h

mFrameErrorRate

uint16_t otMeshDiagChildEntry::mFrameErrorRate

Frame error rate (0x0000->0%, 0xffff->100%).

Definition at line 259 of file include/openthread/mesh_diag.h

mMessageErrorRate

```
uint16_t otMeshDiagChildEntry::mMessageErrorRate
```

(IPv6) msg error rate (0x0000->0%, 0xffff->100%).

Definition at line `260` of file `include/openthread/mesh_diag.h`

mQueuedMessageCount

```
uint16_t otMeshDiagChildEntry::mQueuedMessageCount
```

Number of queued messages for indirect tx to child.

Definition at line `261` of file `include/openthread/mesh_diag.h`

mCslPeriod

```
uint16_t otMeshDiagChildEntry::mCslPeriod
```

CSL Period in unit of 10-symbols-time. Zero indicates CSL is disabled.

Definition at line `262` of file `include/openthread/mesh_diag.h`

mCslTimeout

```
uint32_t otMeshDiagChildEntry::mCslTimeout
```

CSL Timeout in seconds.

Definition at line `263` of file `include/openthread/mesh_diag.h`

mCslChannel

```
uint8_t otMeshDiagChildEntry::mCslChannel
```

CSL channel.

Definition at line `264` of file `include/openthread/mesh_diag.h`

otMeshDiagRouterNeighborEntry

Represents information about a router neighbor entry from `otMeshDiagQueryRouterNeighborTable()`.

`mSupportsErrRate` indicates whether or not the error tracking feature is supported and `mFrameErrorRate` and `mMessageErrorRate` values are valid. The frame error rate tracks frame tx errors (towards the child) at MAC layer, while `mMessageErrorRate` tracks the IPv6 message error rate (above MAC layer and after MAC retries) when an IPv6 message is dropped. For example, if the message is large and requires 6LoWPAN fragmentation, message tx is considered as failed if one of its fragment frame tx fails (for example, never acked).

Public Attributes

bool	mSupportsErrRate `mFrameErrorRate` and `mMessageErrorRate` values are valid.
uint16_t	mRloc16 RLOC16.
otExtAddress	mExtAddress Extended Address.
uint16_t	mVersion Version.
uint32_t	mConnectionTime Seconds since link establishment.
uint8_t	mLinkMargin Link Margin in dB.
int8_t	mAverageRssi Average RSSI.
int8_t	mLastRssi RSSI of last received frame.
uint16_t	mFrameErrorRate Frame error rate (0x0000->0%, 0xffff->100%).
uint16_t	mMessageErrorRate (IPv6) msg error rate (0x0000->0%, 0xffff->100%).

Public Attribute Documentation

mSupportsErrRate

```
bool otMeshDiagRouterNeighborEntry::mSupportsErrRate
```

`mFrameErrorRate` and `mMessageErrorRate` values are valid.

Definition at line `357` of file `include/openthread/mesh_diag.h`

mRloc16

```
uint16_t otMeshDiagRouterNeighborEntry::mRloc16
```

RLOC16.

Definition at line `358` of file `include/openthread/mesh_diag.h`

mExtAddress

```
otExtAddress otMeshDiagRouterNeighborEntry::mExtAddress
```

Extended Address.

Definition at line `359` of file `include/openthread/mesh_diag.h`

mVersion

```
uint16_t otMeshDiagRouterNeighborEntry::mVersion
```

Version.

Definition at line `360` of file `include/openthread/mesh_diag.h`

mConnectionTime

```
uint32_t otMeshDiagRouterNeighborEntry::mConnectionTime
```

Seconds since link establishment.

Definition at line `361` of file `include/openthread/mesh_diag.h`

mLinkMargin

```
uint8_t otMeshDiagRouterNeighborEntry::mLinkMargin
```

Link Margin in dB.

Definition at line `362` of file `include/openthread/mesh_diag.h`

mAverageRssi

```
int8_t otMeshDiagRouterNeighborEntry::mAverageRssi
```

Average RSSI.

Definition at line `363` of file `include/openthread/mesh_diag.h`

mLastRssi

int8_t otMeshDiagRouterNeighborEntry::mLastRssi

RSSI of last received frame.

Definition at line 364 of file include/openthread/mesh_diag.h

mFrameErrorRate

uint16_t otMeshDiagRouterNeighborEntry::mFrameErrorRate

Frame error rate (0x0000->0%, 0xffff->100%).

Definition at line 365 of file include/openthread/mesh_diag.h

mMessageErrorRate

uint16_t otMeshDiagRouterNeighborEntry::mMessageErrorRate

(IPv6) msg error rate (0x0000->0%, 0xffff->100%).

Definition at line 366 of file include/openthread/mesh_diag.h

Network Co-Processor

Network Co-Processor

This module includes functions that control the Thread stack's execution.

Typedefs

typedef int(* otNcpHdlcSendCallback)(const uint8_t *aBuf, uint16_t aBufLength)
Pointer is called to send HDLC encoded NCP data.

typedef bool(* otNcpDelegateAllowPeekPoke)(uint32_t aAddress, uint16_t aCount)
Defines delegate (function pointer) type to control behavior of peek/poke operation.

Functions

void otNcpHdlcSendDone(void)
Is called after NCP send finished.

void otNcpHdlcReceive(const uint8_t *aBuf, uint16_t aBufLength)
Is called after HDLC encoded NCP data received.

void otNcpHdlcInit(otInstance *aInstance, otNcpHdlcSendCallback aSendCallback)
Initialize the NCP based on HDLC framing.

void otNcpSpiInit(otInstance *aInstance)
Initialize the NCP based on SPI framing.

otError otNcpStreamWrite(int aStreamId, const uint8_t *aDataPtr, int aDataLen)
Send data to the host via a specific stream.

void otNcpPlatLogv(otLogLevel aLogLevel, otLogRegion aLogRegion, const char *aFormat, va_list aArgs)
Writes OpenThread Log using `otNcpStreamWrite`.

void otNcpRegisterPeekPokeDelegates(otNcpDelegateAllowPeekPoke aAllowPeekDelegate, otNcpDelegateAllowPeekPoke aAllowPokeDelegate)
Registers peek/poke delegate functions with NCP module.

Typedef Documentation

otNcpHdlcSendCallback

```
typedef int(* otNcpHdlcSendCallback) (const uint8_t *aBuf, uint16_t aBufLength) )(const uint8_t *aBuf, uint16_t aBufLength)
```

Pointer is called to send HDLC encoded NCP data.

Parameters

[in]	aBuf	A pointer to a buffer with an output.
[in]	aBufLength	A length of the output data stored in the buffer.

Returns

Number of bytes processed by the callback.

Definition at line 66 of file include/openthread/ncp.h

otNcpDelegateAllowPeekPoke

```
typedef bool(* otNcpDelegateAllowPeekPoke) (uint32_t aAddress, uint16_t aCount) )(uint32_t aAddress, uint16_t aCount)
```

Defines delegate (function pointer) type to control behavior of peek/poke operation.

Parameters

[in]	aAddress	Start address of memory region.
[in]	aCount	Number of bytes to peek or poke.

This delegate function is called to decide whether to allow peek or poke of a specific memory region. It is used if NCP support for peek/poke commands is enabled.

Returns

- TRUE to allow peek/poke of the given memory region, FALSE otherwise.

Definition at line 149 of file include/openthread/ncp.h

Function Documentation

otNcpHdlcSendDone

```
void otNcpHdlcSendDone (void)
```

Is called after NCP send finished.

Parameters

N/A		

Definition at line 72 of file include/openthread/ncp.h

otNcpHdlcReceive

```
void otNcpHdlcReceive (const uint8_t *aBuf, uint16_t aBufLength)
```

Is called after HDLC encoded NCP data received.

Parameters

[in]	aBuf	A pointer to a buffer.
[in]	aBufLength	The length of the data stored in the buffer.

Definition at line 81 of file include/openthread/ncp.h

otNcpHdlcInit

```
void otNcpHdlcInit (otInstance *aInstance, otNcpHdlcSendCallback aSendCallback)
```

Initialize the NCP based on HDLC framing.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aSendCallback	The function pointer used to send NCP data.

Definition at line `90` of file `include/openthread/ncp.h`

otNcpSpiInit

```
void otNcpSpiInit (otInstance *aInstance)
```

Initialize the NCP based on SPI framing.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `98` of file `include/openthread/ncp.h`

otNcpStreamWrite

```
otError otNcpStreamWrite (int aStreamId, const uint8_t *aDataPtr, int aDataLen)
```

Send data to the host via a specific stream.

Parameters

[in]	aStreamId	A numeric identifier for the stream to write to. If set to '0', will default to the debug stream.
[in]	aDataPtr	A pointer to the data to send on the stream. If aDataLen is non-zero, this param MUST NOT be NULL.
[in]	aDataLen	The number of bytes of data from aDataPtr to send.

Attempts to send the given data to the host using the given aStreamId. This is useful for reporting error messages, implementing debug/diagnostic consoles, and potentially other types of datastreams.

The write either is accepted in its entirety or rejected. Partial writes are not attempted.

Definition at line `122` of file `include/openthread/ncp.h`

otNcpPlatLogv

```
void otNcpPlatLogv (otLogLevel aLogLevel, otLogRegion aLogRegion, const char *aFormat, va_list aArgs)
```

Writes OpenThread Log using `otNcpStreamWrite` .

Parameters

[in]	aLogLevel	The log level.
[in]	aLogRegion	The log region.
[in]	aFormat	A pointer to the format string.
[in]	aArgs	va_list matching aFormat.

Definition at line `132` of file `include/openthread/ncp.h`

otNcpRegisterPeekPokeDelegates

> void otNcpRegisterPeekPokeDelegates (otNcpDelegateAllowPeekPoke aAllowPeekDelegate, otNcpDelegateAllowPeekPoke aAllowPokeDelegate)

Registers peek/poke delegate functions with NCP module.

Parameters

[in]	aAllowPeekDelegate	Delegate function pointer for peek operation.
[in]	aAllowPokeDelegate	Delegate function pointer for poke operation.

The delegate functions are called by NCP module to decide whether to allow peek or poke of a specific memory region. If the delegate pointer is set to NULL, it allows peek/poke operation for any address.

Definition at line `161` of file `include/openthread/ncp.h`

Network Time Synchronization

Network Time Synchronization

This module includes functions that control network time synchronization service.

Enumerations

enum otNetworkTimeStatus {

OT_NETWORK_TIME_UNSYNCHRONIZED = -1
OT_NETWORK_TIME_RESYNC_NEEDED = 0
OT_NETWORK_TIME_SYNCHRONIZED = 1

}

Represents OpenThread time synchronization status.

Typedefs

typedef enum otNetworkTimeStatus

Represents OpenThread time synchronization status.

typedef void(* otNetworkTimeSyncCallbackFn)(void *aCallbackContext)

Pointer is called when a network time sync or status change occurs.

Functions

otNetworkTimeStatus otNetworkTimeGet(otInstance *aInstance, uint64_t *aNetworkTime)

Get the Thread network time.

otError otNetworkTimeSetSyncPeriod(otInstance *aInstance, uint16_t aTimeSyncPeriod)

Set the time synchronization period.

uint16_t otNetworkTimeGetSyncPeriod(otInstance *aInstance)

Get the time synchronization period.

otError otNetworkTimeSetXtalThreshold(otInstance *aInstance, uint16_t aXTALThreshold)

Set the time synchronization XTAL accuracy threshold for Router-Capable device.

uint16_t otNetworkTimeGetXtalThreshold(otInstance *aInstance)

Get the time synchronization XTAL accuracy threshold for Router.

void otNetworkTimeSyncSetCallback(otInstance *aInstance, otNetworkTimeSyncCallbackFn aCallbackFn, void *aCallbackContext)

Set a callback to be called when a network time sync or status change occurs.

Macros

#define OT_TIME_SYNC_INVALID_SEQ 0

zero is considered as invalid time synchronization sequence.

Enumeration Documentation

otNetworkTimeStatus

> otNetworkTimeStatus

Represents OpenThread time synchronization status.

Enumerator	
OT_NETWORK_TIME_UNSYNCHRONIZED	The device hasn't attached to a network.
OT_NETWORK_TIME_RESYNC_NEEDED	The device hasn't received time sync for more than two periods time.
OT_NETWORK_TIME_SYNCHRONIZED	The device network time is synchronized.

Definition at line `59` of file `include/openthread/network_time.h`

Typedef Documentation

otNetworkTimeStatus

> typedef enum otNetworkTimeStatus otNetworkTimeStatus

Represents OpenThread time synchronization status.

Definition at line `64` of file `include/openthread/network_time.h`

otNetworkTimeSyncCallbackFn

> typedef void(* otNetworkTimeSyncCallbackFn) (void *aCallbackContext))(void *aCallbackContext)

Pointer is called when a network time sync or status change occurs.

Definition at line `70` of file `include/openthread/network_time.h`

Function Documentation

otNetworkTimeGet

> otNetworkTimeStatus otNetworkTimeGet (otInstance *aInstance, uint64_t *aNetworkTime)

Get the Thread network time.

Parameters

[in]	aInstance	The OpenThread instance structure.
[inout]	aNetworkTime	The Thread network time in microseconds.

Returns

- The time synchronization status.

Definition at line `87` of file `include/openthread/network_time.h`

otNetworkTimeSetSyncPeriod

```
otError otNetworkTimeSetSyncPeriod (otInstance *aInstance, uint16_t aTimeSyncPeriod)
```

Set the time synchronization period.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aTimeSyncPeriod	The time synchronization period, in seconds.

Can only be called while Thread protocols are disabled.

Definition at line 101 of file include/openthread/network_time.h

otNetworkTimeGetSyncPeriod

```
uint16_t otNetworkTimeGetSyncPeriod (otInstance *aInstance)
```

Get the time synchronization period.

Parameters

[in]	aInstance	The OpenThread instance structure.

Returns

- The time synchronization period.

Definition at line 111 of file include/openthread/network_time.h

otNetworkTimeSetXtalThreshold

```
otError otNetworkTimeSetXtalThreshold (otInstance *aInstance, uint16_t aXTALThreshold)
```

Set the time synchronization XTAL accuracy threshold for Router-Capable device.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aXTALThreshold	The XTAL accuracy threshold for Router, in PPM.

Can only be called while Thread protocols are disabled.

Definition at line 125 of file include/openthread/network_time.h

otNetworkTimeGetXtalThreshold

```
uint16_t otNetworkTimeGetXtalThreshold (otInstance *aInstance)
```

Get the time synchronization XTAL accuracy threshold for Router.

Parameters

[in]	aInstance	The OpenThread instance structure.

Returns

- The XTAL accuracy threshold for Router, in PPM.

Definition at line 135 of file include/openthread/network_time.h

otNetworkTimeSyncSetCallback

```
void otNetworkTimeSyncSetCallback (otInstance *aInstance, otNetworkTimeSyncCallbackFn aCallbackFn, void *aCallbackContext)
```

Set a callback to be called when a network time sync or status change occurs.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aCallbackFn	The callback function to be called
[in]	aCallbackContext	The context to be passed to the callback function upon invocation

This callback shall be called only when the network time offset jumps by OPENTHREAD_CONFIG_TIME_SYNC_JUMP_NOTIF_MIN_US or when the status changes.

Definition at line 148 of file include/openthread/network_time.h

Macro Definition Documentation

OT_TIME_SYNC_INVALID_SEQ

```
#define OT_TIME_SYNC_INVALID_SEQ
```

Value:

```
0
```

zero is considered as invalid time synchronization sequence.

Definition at line 76 of file include/openthread/network_time.h

Radio Statistics

Radio Statistics

This module includes functions for radio statistics.

Modules

otRadioTimeStats

Typedefs

typedef struct otRadioTimeStats	otRadioTimeStats Contains the statistics of radio.

Functions

const otRadioTimeStats *	otRadioTimeStatsGet(otInstance *aInstance) Gets the radio statistics.
void	otRadioTimeStatsReset(otInstance *aInstance) Resets the radio statistics.

Typedef Documentation

otRadioTimeStats

```
typedef struct otRadioTimeStats otRadioTimeStats
```

Contains the statistics of radio.

Definition at line 67 of file include/openthread/radio_stats.h

Function Documentation

otRadioTimeStatsGet

```
const otRadioTimeStats * otRadioTimeStatsGet (otInstance *aInstance)
```

Gets the radio statistics.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The radio statistics include the time when the radio is in TX/RX/Sleep state. These times are in units of microseconds. All times are calculated from the last reset of radio statistics.

Returns

A const pointer to the otRadioTimeStats struct that contains the data.

Definition at line 80 of file include/openthread/radio_stats.h

otRadioTimeStatsReset

```
void otRadioTimeStatsReset (otInstance *aInstance)
```

Resets the radio statistics.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

All times are reset to 0.

Definition at line 91 of file include/openthread/radio_stats.h

otRadioTimeStats

Contains the statistics of radio.

Public Attributes

uint64_t	mDisabledTime The total time that radio is in disabled state, in unit of microseconds.
uint64_t	mSleepTime The total time that radio is in sleep state, in unit of microseconds.
uint64_t	mTxTime
uint64_t	mRxTime

> The total time that radio is doing transmission, in unit of microseconds.

Public Attribute Documentation

mDisabledTime

```
uint64_t otRadioTimeStats::mDisabledTime
```

The total time that radio is in disabled state, in unit of microseconds.

Definition at line `63` of file `include/openthread/radio_stats.h`

mSleepTime

```
uint64_t otRadioTimeStats::mSleepTime
```

The total time that radio is in sleep state, in unit of microseconds.

Definition at line `64` of file `include/openthread/radio_stats.h`

mTxTime

```
uint64_t otRadioTimeStats::mTxTime
```

Definition at line `65` of file `include/openthread/radio_stats.h`

mRxTime

```
uint64_t otRadioTimeStats::mRxTime
```

> The total time that radio is doing transmission, in unit of microseconds.

Definition at line 66 of file include/openthread/radio_stats.h

Random Number Generator

Random Number Generator

Modules

RNG Cryptographic

RNG Non-cryptographic

RNG Cryptographic

RNG Cryptographic

This module includes functions that generates cryptographic random numbers.

Functions

otError otRandomCryptoFillBuffer(uint8_t *aBuffer, uint16_t aSize)
Fills a given buffer with cryptographically secure random bytes.

Function Documentation

otRandomCryptoFillBuffer

> otError otRandomCryptoFillBuffer (uint8_t *aBuffer, uint16_t aSize)

Fills a given buffer with cryptographically secure random bytes.

Parameters

[out]	aBuffer	A pointer to a buffer to fill with the random bytes.
[in]	aSize	Size of buffer (number of bytes to fill).

Definition at line `63` of file `include/openthread/random_crypto.h`

RNG Non-cryptographic

RNG Non-cryptographic

This module includes functions that generates non cryptographic random numbers.

Functions

uint32_t otRandomNonCryptoGetUint32(void)
Generates and returns a random `uint32_t` value.

uint8_t otRandomNonCryptoGetUint8(void)
Generates and returns a random byte.

uint16_t otRandomNonCryptoGetUint16(void)
Generates and returns a random `uint16_t` value.

uint8_t otRandomNonCryptoGetUint8InRange(uint8_t aMin, uint8_t aMax)
Generates and returns a random `uint8_t` value within a given range `[aMin, aMax)`.

uint16_t otRandomNonCryptoGetUint16InRange(uint16_t aMin, uint16_t aMax)
Generates and returns a random `uint16_t` value within a given range `[aMin, aMax)`.

uint32_t otRandomNonCryptoGetUint32InRange(uint32_t aMin, uint32_t aMax)
Generates and returns a random `uint32_t` value within a given range `[aMin, aMax)`.

void otRandomNonCryptoFillBuffer(uint8_t *aBuffer, uint16_t aSize)
Fills a given buffer with random bytes.

uint32_t otRandomNonCryptoAddJitter(uint32_t aValue, uint16_t aJitter)
Adds a random jitter within a given range to a given value.

Function Documentation

otRandomNonCryptoGetUint32

```
uint32_t otRandomNonCryptoGetUint32 (void)
```

Generates and returns a random `uint32_t` value.

Parameters

N/A

Returns

- A random `uint32_t` value.

Definition at line `60` of file `include/openthread/random_noncrypto.h`

otRandomNonCryptoGetUint8

```
uint8_t otRandomNonCryptoGetUint8 (void)
```

Generates and returns a random byte.

Parameters

N/A		

Returns

- A random `uint8_t` value.

Definition at line `68` of file `include/openthread/random_noncrypto.h`

otRandomNonCryptoGetUint16

```
uint16_t otRandomNonCryptoGetUint16 (void)
```

Generates and returns a random `uint16_t` value.

Parameters

N/A		

Returns

- A random `uint16_t` value.

Definition at line `76` of file `include/openthread/random_noncrypto.h`

otRandomNonCryptoGetUint8InRange

```
uint8_t otRandomNonCryptoGetUint8InRange (uint8_t aMin, uint8_t aMax)
```

Generates and returns a random `uint8_t` value within a given range `[aMin, aMax)`.

Parameters

[in]	aMin	A minimum value (this value can be included in returned random result).
[in]	aMax	A maximum value (this value is excluded from returned random result).

Returns

- A random `uint8_t` value in the given range (i.e., aMin <= random value < aMax).

Definition at line `86` of file `include/openthread/random_noncrypto.h`

otRandomNonCryptoGetUint16InRange

```
uint16_t otRandomNonCryptoGetUint16InRange (uint16_t aMin, uint16_t aMax)
```

Generates and returns a random `uint16_t` value within a given range `[aMin, aMax)`.

Parameters

[in]	aMin	A minimum value (this value can be included in returned random result).
[in]	aMax	A maximum value (this value is excluded from returned random result).

Note

The returned random value can include the `aMin` value but excludes the `aMax`.

Returns

- A random `uint16_t` value in the given range (i.e., aMin <= random value < aMax).

Definition at line `98` of file `include/openthread/random_noncrypto.h`

otRandomNonCryptoGetUint32InRange

```
uint32_t otRandomNonCryptoGetUint32InRange (uint32_t aMin, uint32_t aMax)
```

Generates and returns a random `uint32_t` value within a given range `[aMin, aMax)`.

Parameters

[in]	aMin	A minimum value (this value can be included in returned random result).
[in]	aMax	A maximum value (this value is excluded from returned random result).

Note

- The returned random value can include the `aMin` value but excludes the `aMax`.

Returns

- A random `uint32_t` value in the given range (i.e., aMin <= random value < aMax).

Definition at line `111` of file `include/openthread/random_noncrypto.h`

otRandomNonCryptoFillBuffer

```
void otRandomNonCryptoFillBuffer (uint8_t *aBuffer, uint16_t aSize)
```

Fills a given buffer with random bytes.

Parameters

[out]	aBuffer	A pointer to a buffer to fill with the random bytes.
[in]	aSize	Size of buffer (number of bytes to fill).

Definition at line `120` of file `include/openthread/random_noncrypto.h`

otRandomNonCryptoAddJitter

```
uint32_t otRandomNonCryptoAddJitter (uint32_t aValue, uint16_t aJitter)
```

Adds a random jitter within a given range to a given value.

Parameters

[in]	aValue	A value to which the random jitter is added.
[in]	aJitter	Maximum jitter. Random jitter is selected from the range `[-aJitter, aJitter]`.

Returns

- The given value with an added random jitter.

Definition at line 131 of file include/openthread/random_noncrypto.h

SNTP

SNTP

This module includes functions that control SNTP communication.

Modules

otSntpQuery

Typedefs

typedef struct otSntpQuery	otSntpQuery Implements SNTP Query parameters.
typedef void(*	otSntpResponseHandler)(void *aContext, uint64_t aTime, otError aResult) Pointer is called when a SNTP response is received.

Functions

otError	otSntpClientQuery(otInstance *aInstance, const otSntpQuery *aQuery, otSntpResponseHandler aHandler, void *aContext) Sends a SNTP query.
void	otSntpClientSetUnixEra(otInstance *aInstance, uint32_t aUnixEra) Sets the unix era number.

Macros

#define	OT_SNTP_DEFAULT_SERVER_IP "2001:4860:4806:8::" Defines default SNTP Server address - Google NTP Server.
#define	OT_SNTP_DEFAULT_SERVER_PORT 123 Defines default SNTP Server port.

Typedef Documentation

otSntpQuery

```
typedef struct otSntpQuery otSntpQuery
```

Implements SNTP Query parameters.

Definition at line 66 of file include/openthread/sntp.h

otSntpResponseHandler

```
typedef void(* otSntpResponseHandler) (void *aContext, uint64_t aTime, otError aResult) )(void *aContext, uint64_t aTime, otError aResult)
```

Pointer is called when a SNTP response is received.

Parameters

[in]	aContext	A pointer to application-specific context.
[in]	aTime	Specifies the time at the server when the response left for the client, in UNIX time.
[in]	aResult	A result of the SNTP transaction.

Definition at line `83` of file `include/openthread/sntp.h`

Function Documentation

otSntpClientQuery

```
otError otSntpClientQuery (otInstance *aInstance, const otSntpQuery *aQuery, otSntpResponseHandler aHandler, void *aContext)
```

Sends a SNTP query.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aQuery	A pointer to specify SNTP query parameters.
[in]	aHandler	A function pointer that shall be called on response reception or time-out.
[in]	aContext	A pointer to arbitrary context information.

Is available only if feature `OPENTHREAD_CONFIG_SNTP_CLIENT_ENABLE` is enabled.

Definition at line `96` of file `include/openthread/sntp.h`

otSntpClientSetUnixEra

```
void otSntpClientSetUnixEra (otInstance *aInstance, uint32_t aUnixEra)
```

Sets the unix era number.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aUnixEra	Unix era number.

The default value of unix era is set to 0. The subsequent eras start after year 2106.

Definition at line `110` of file `include/openthread/sntp.h`

Macro Definition Documentation

OT_SNTP_DEFAULT_SERVER_IP

```
#define OT_SNTP_DEFAULT_SERVER_IP
```

Value:

```
"2001:4860:4806:8::"
```

Defines default SNTP Server address - Google NTP Server.

Definition at line 56 of file include/openthread/sntp.h

OT_SNTP_DEFAULT_SERVER_PORT

```
#define OT_SNTP_DEFAULT_SERVER_PORT
```

Value:

```
123
```

Defines default SNTP Server port.

Definition at line 57 of file include/openthread/sntp.h

otSntpQuery

Implements SNTP Query parameters.

Public Attributes

const otMessageInfo *	mMessageInfo A reference to the message info related with SNTP Server.

Public Attribute Documentation

mMessageInfo

```
const otMessageInfo* otSntpQuery::mMessageInfo
```

A reference to the message info related with SNTP Server.

Definition at line 65 of file include/openthread/sntp.h

Platform Abstraction

Platform Abstraction

This module includes the platform abstraction used by the OpenThread stack.

Modules

Alarm

Crypto - Platform

DNS - Platform

Entropy

Factory Diagnostics - Platform

Logging - Platform

Memory

Message Pool

Miscellaneous

Network Simulator

Radio

Settings

SPI Slave

Time Service

Toolchain

TREL - Platform

Infrastructure Interface

Alarm

Alarm

This module includes the platform abstraction for the alarm service.

Functions

void otPlatAlarmMicroStartAt(otInstance *aInstance, uint32_t aT0, uint32_t aDt)
Set the alarm to fire at `aDt` microseconds after `aT0`.

void otPlatAlarmMicroStop(otInstance *aInstance)
Stop the alarm.

uint32_t otPlatAlarmMicroGetNow(void)
Get the current time.

void otPlatAlarmMicroFired(otInstance *aInstance)
Signal that the alarm has fired.

void otPlatAlarmMilliStartAt(otInstance *aInstance, uint32_t aT0, uint32_t aDt)
Set the alarm to fire at `aDt` milliseconds after `aT0`.

void otPlatAlarmMilliStop(otInstance *aInstance)
Stop the alarm.

uint32_t otPlatAlarmMilliGetNow(void)
Get the current time.

void otPlatAlarmMilliFired(otInstance *aInstance)
Signal that the alarm has fired.

void otPlatDiagAlarmFired(otInstance *aInstance)
Signal diagnostics module that the alarm has fired.

Function Documentation

otPlatAlarmMicroStartAt

> void otPlatAlarmMicroStartAt (otInstance *aInstance, uint32_t aT0, uint32_t aDt)

Set the alarm to fire at `aDt` microseconds after `aT0`.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aT0	The reference time.
[in]	aDt	The time delay in microseconds from `aT0`.

For `aT0`, the platform MUST support all values in [0, 2^32-1]. For `aDt`, the platform MUST support all values in [0, 2^31-1].

Definition at line `64` of file `include/openthread/platform/alarm-micro.h`

otPlatAlarmMicroStop

```
void otPlatAlarmMicroStop (otInstance *aInstance)
```

Stop the alarm.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `72` of file `include/openthread/platform/alarm-micro.h`

otPlatAlarmMicroGetNow

```
uint32_t otPlatAlarmMicroGetNow (void)
```

Get the current time.

Parameters

N/A		

The current time MUST represent a free-running timer. When maintaining current time, the time value MUST utilize the entire range [0, 2^32-1] and MUST NOT wrap before 2^32.

Returns

- The current time in microseconds.

Definition at line `83` of file `include/openthread/platform/alarm-micro.h`

otPlatAlarmMicroFired

```
void otPlatAlarmMicroFired (otInstance *aInstance)
```

Signal that the alarm has fired.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `90` of file `include/openthread/platform/alarm-micro.h`

otPlatAlarmMilliStartAt

```
void otPlatAlarmMilliStartAt (otInstance *aInstance, uint32_t aT0, uint32_t aDt)
```

Set the alarm to fire at `aDt` milliseconds after `aT0`.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aT0	The reference time.
[in]	aDt	The time delay in milliseconds from `aT0`.

For `aT0` the platform MUST support all values in [0, 2^32-1]. For `aDt`, the platform MUST support all values in [0, 2^31-1].

Definition at line 66 of file include/openthread/platform/alarm-milli.h

otPlatAlarmMilliStop

```
void otPlatAlarmMilliStop (otInstance *aInstance)
```

Stop the alarm.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line 73 of file include/openthread/platform/alarm-milli.h

otPlatAlarmMilliGetNow

```
uint32_t otPlatAlarmMilliGetNow (void)
```

Get the current time.

Parameters

N/A		

The current time MUST represent a free-running timer. When maintaining current time, the time value MUST utilize the entire range [0, 2^32-1] and MUST NOT wrap before 2^32.

Returns

- The current time in milliseconds.

Definition at line 83 of file include/openthread/platform/alarm-milli.h

otPlatAlarmMilliFired

```
void otPlatAlarmMilliFired (otInstance *aInstance)
```

Signal that the alarm has fired.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line 90 of file include/openthread/platform/alarm-milli.h

otPlatDiagAlarmFired

```
void otPlatDiagAlarmFired (otInstance *aInstance)
```

Signal diagnostics module that the alarm has fired.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line 97 of file include/openthread/platform/alarm-milli.h

Crypto - Platform

Crypto - Platform

This module includes the platform abstraction for Crypto.

Modules

otCryptoKey

otCryptoContext

otPlatCryptoSha256Hash

otPlatCryptoEcdsaKeyPair

otPlatCryptoEcdsaPublicKey

otPlatCryptoEcdsaSignature

Enumerations

enum otCryptoKeyType {

OT_CRYPTO_KEY_TYPE_RAW
OT_CRYPTO_KEY_TYPE_AES
OT_CRYPTO_KEY_TYPE_HMAC
OT_CRYPTO_KEY_TYPE_ECDSA

}

Defines the key types.

enum otCryptoKeyAlgorithm {

OT_CRYPTO_KEY_ALG_VENDOR
OT_CRYPTO_KEY_ALG_AES_ECB
OT_CRYPTO_KEY_ALG_HMAC_SHA_256
OT_CRYPTO_KEY_ALG_ECDSA

}

Defines the key algorithms.

enum @11 {

OT_CRYPTO_KEY_USAGE_NONE = 0
OT_CRYPTO_KEY_USAGE_EXPORT = 1 << 0
OT_CRYPTO_KEY_USAGE_ENCRYPT = 1 << 1
OT_CRYPTO_KEY_USAGE_DECRYPT = 1 << 2
OT_CRYPTO_KEY_USAGE_SIGN_HASH = 1 << 3
OT_CRYPTO_KEY_USAGE_VERIFY_HASH = 1 << 4

}

Defines the key usage flags.

enum otCryptoKeyStorage {

OT_CRYPTO_KEY_STORAGE_VOLATILE
OT_CRYPTO_KEY_STORAGE_PERSISTENT

}

Defines the key storage types.

typedef uint32_t	otCryptoKeyRef This datatype represents the key reference.
typedef struct otCryptoKey	otCryptoKey
typedef struct otCryptoContext	otCryptoContext
typedef struct otPlatCryptoSha256Hash	otPlatCryptoSha256Hash Represents a SHA-256 hash.
typedef struct otPlatCryptoEcdsaKeyPair	otPlatCryptoEcdsaKeyPair
typedef struct otPlatCryptoEcdsaPublicKey	otPlatCryptoEcdsaPublicKey
typedef struct otPlatCryptoEcdsaSignature	otPlatCryptoEcdsaSignature

Variables

OT_TOOL_PACKED_BEGIN struct otPlatCryptoSha256Hash	OT_TOOL_PACKED_END

Functions

void	otPlatCryptoInit(void) Initialize the Crypto module.
otError	otPlatCryptoImportKey(otCryptoKeyRef *aKeyRef, otCryptoKeyType aKeyType, otCryptoKeyAlgorithm aKeyAlgorithm, int aKeyUsage, otCryptoKeyStorage aKeyPersistence, const uint8_t *aKey, size_t aKeyLen) Import a key into PSA ITS.
otError	otPlatCryptoExportKey(otCryptoKeyRef aKeyRef, uint8_t *aBuffer, size_t aBufferLen, size_t *aKeyLen) Export a key stored in PSA ITS.
otError	otPlatCryptoDestroyKey(otCryptoKeyRef aKeyRef) Destroy a key stored in PSA ITS.
bool	otPlatCryptoHasKey(otCryptoKeyRef aKeyRef) Check if the key ref passed has an associated key in PSA ITS.
otError	otPlatCryptoHmacSha256Init(otCryptoContext *aContext) Initialize the HMAC operation.
otError	otPlatCryptoHmacSha256Deinit(otCryptoContext *aContext) Uninitialize the HMAC operation.
otError	otPlatCryptoHmacSha256Start(otCryptoContext *aContext, const otCryptoKey *aKey) Start HMAC operation.
otError	otPlatCryptoHmacSha256Update(otCryptoContext *aContext, const void *aBuf, uint16_t aBufLength) Update the HMAC operation with new input.

otError otPlatCryptoHmacSha256Finish(otCryptoContext *aContext, uint8_t *aBuf, size_t aBufLength)
Complete the HMAC operation.

otError otPlatCryptoAesInit(otCryptoContext *aContext)
Initialise the AES operation.

otError otPlatCryptoAesSetKey(otCryptoContext *aContext, const otCryptoKey *aKey)
Set the key for AES operation.

otError otPlatCryptoAesEncrypt(otCryptoContext *aContext, const uint8_t *aInput, uint8_t *aOutput)
Encrypt the given data.

otError otPlatCryptoAesFree(otCryptoContext *aContext)
Free the AES context.

otError otPlatCryptoHkdfInit(otCryptoContext *aContext)
Initialise the HKDF context.

otError otPlatCryptoHkdfExpand(otCryptoContext *aContext, const uint8_t *aInfo, uint16_t aInfoLength, uint8_t *aOutputKey, uint16_t aOutputKeyLength)
Perform HKDF Expand step.

otError otPlatCryptoHkdfExtract(otCryptoContext *aContext, const uint8_t *aSalt, uint16_t aSaltLength, const otCryptoKey *aInputKey)
Perform HKDF Extract step.

otError otPlatCryptoHkdfDeinit(otCryptoContext *aContext)
Uninitialize the HKDF context.

otError otPlatCryptoSha256Init(otCryptoContext *aContext)
Initialise the SHA-256 operation.

otError otPlatCryptoSha256Deinit(otCryptoContext *aContext)
Uninitialize the SHA-256 operation.

otError otPlatCryptoSha256Start(otCryptoContext *aContext)
Start SHA-256 operation.

otError otPlatCryptoSha256Update(otCryptoContext *aContext, const void *aBuf, uint16_t aBufLength)
Update SHA-256 operation with new input.

otError otPlatCryptoSha256Finish(otCryptoContext *aContext, uint8_t *aHash, uint16_t aHashSize)
Finish SHA-256 operation.

void otPlatCryptoRandomInit(void)
Initialize cryptographically-secure pseudorandom number generator (CSPRNG).

void otPlatCryptoRandomDeinit(void)
Deinitialize cryptographically-secure pseudorandom number generator (CSPRNG).

otError otPlatCryptoRandomGet(uint8_t *aBuffer, uint16_t aSize)
Fills a given buffer with cryptographically secure random bytes.

otError otPlatCryptoEcdsaGenerateKey(otPlatCryptoEcdsaKeyPair *aKeyPair)
Generate and populate the output buffer with a new ECDSA key-pair.

otError otPlatCryptoEcdsaGetPublicKey(const otPlatCryptoEcdsaKeyPair *aKeyPair, otPlatCryptoEcdsaPublicKey *aPublicKey)
Get the associated public key from the input context.

otError otPlatCryptoEcdsaSign(const otPlatCryptoEcdsaKeyPair *aKeyPair, const otPlatCryptoSha256Hash *aHash, otPlatCryptoEcdsaSignature *aSignature)
Calculate the ECDSA signature for a hashed message using the private key from the input context.

otError otPlatCryptoEcdsaVerify(const otPlatCryptoEcdsaPublicKey *aPublicKey, const otPlatCryptoSha256Hash *aHash, const otPlatCryptoEcdsaSignature *aSignature)
Use the key from the input context to verify the ECDSA signature of a hashed message.

otError otPlatCryptoEcdsaSignUsingKeyRef(otCryptoKeyRef aKeyRef, const otPlatCryptoSha256Hash *aHash, otPlatCryptoEcdsaSignature *aSignature)
Calculate the ECDSA signature for a hashed message using the Key reference passed.

otError otPlatCryptoEcdsaExportPublicKey(otCryptoKeyRef aKeyRef, otPlatCryptoEcdsaPublicKey *aPublicKey)
Get the associated public key from the key reference passed.

otError otPlatCryptoEcdsaGenerateAndImportKey(otCryptoKeyRef aKeyRef)
Generate and import a new ECDSA key-pair at reference passed.

otError otPlatCryptoEcdsaVerifyUsingKeyRef(otCryptoKeyRef aKeyRef, const otPlatCryptoSha256Hash *aHash, const otPlatCryptoEcdsaSignature *aSignature)
Use the keyref to verify the ECDSA signature of a hashed message.

void otPlatCryptoPbkdf2GenerateKey(const uint8_t *aPassword, uint16_t aPasswordLen, const uint8_t *aSalt, uint16_t aSaltLen, uint32_t alterationCounter, uint16_t aKeyLen, uint8_t *aKey)
Perform PKCS#5 PBKDF2 using CMAC (AES-CMAC-PRF-128).

Macros

#define OT_CRYPTO_SHA256_HASH_SIZE 32
Length of SHA256 hash (in bytes).

#define OT_CRYPTO_ECDSA_MAX_DER_SIZE 125
Max buffer size (in bytes) for representing the EDCSA key-pair in DER format.

#define OT_CRYPTO_ECDSA_PUBLIC_KEY_SIZE 64
Buffer size (in bytes) for representing the EDCSA public key.

#define OT_CRYPTO_ECDSA_SIGNATURE_SIZE 64
Buffer size (in bytes) for representing the EDCSA signature.

#define OT_CRYPTO_PBDKF2_MAX_SALT_SIZE 30
Max PBKDF2 SALT length: salt prefix (6) + extended panid (8) + network name (16)

Enumeration Documentation

otCryptoKeyType

otCryptoKeyType

Defines the key types.

Enumerator	
OT_CRYPTO_KEY_TYPE_RAW	Key Type: Raw Data.
OT_CRYPTO_KEY_TYPE_AES	Key Type: AES.
OT_CRYPTO_KEY_TYPE_HMAC	Key Type: HMAC.
OT_CRYPTO_KEY_TYPE_ECDSA	Key Type: ECDSA.

Definition at line 61 of file include/openthread/platform/crypto.h

otCryptoKeyAlgorithm

otCryptoKeyAlgorithm

Defines the key algorithms.

Enumerator	
OT_CRYPTO_KEY_ALG_VENDOR	Key Algorithm: Vendor Defined.
OT_CRYPTO_KEY_ALG_AES_ECB	Key Algorithm: AES ECB.
OT_CRYPTO_KEY_ALG_HMAC_SHA_256	Key Algorithm: HMAC SHA-256.
OT_CRYPTO_KEY_ALG_ECDSA	Key Algorithm: ECDSA.

Definition at line `73` of file `include/openthread/platform/crypto.h`

@11

> @11

Defines the key usage flags.

Enumerator	
OT_CRYPTO_KEY_USAGE_NONE	Key Usage: Key Usage is empty.
OT_CRYPTO_KEY_USAGE_EXPORT	Key Usage: Key can be exported.
OT_CRYPTO_KEY_USAGE_ENCRYPT	Key Usage: Encryption (vendor defined).
OT_CRYPTO_KEY_USAGE_DECRYPT	Key Usage: AES ECB.
OT_CRYPTO_KEY_USAGE_SIGN_HASH	Key Usage: Sign Hash.
OT_CRYPTO_KEY_USAGE_VERIFY_HASH	Key Usage: Verify Hash.

Definition at line `85` of file `include/openthread/platform/crypto.h`

otCryptoKeyStorage

> otCryptoKeyStorage

Defines the key storage types.

Enumerator	
OT_CRYPTO_KEY_STORAGE_VOLATILE	Key Persistence: Key is volatile.
OT_CRYPTO_KEY_STORAGE_PERSISTENT	Key Persistence: Key is persistent.

Definition at line `99` of file `include/openthread/platform/crypto.h`

Typedef Documentation

otCryptoKeyRef

> typedef uint32_t otCryptoKeyRef

This datatype represents the key reference.

Definition at line `109` of file `include/openthread/platform/crypto.h`

otCryptoKey

```
typedef struct otCryptoKey otCryptoKey
```

Definition at line 122 of file include/openthread/platform/crypto.h

otCryptoContext

```
typedef struct otCryptoContext otCryptoContext
```

Definition at line 134 of file include/openthread/platform/crypto.h

otPlatCryptoSha256Hash

```
typedef struct otPlatCryptoSha256Hash otPlatCryptoSha256Hash
```

Represents a SHA-256 hash.

Definition at line 158 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaKeyPair

```
typedef struct otPlatCryptoEcdsaKeyPair otPlatCryptoEcdsaKeyPair
```

Definition at line 178 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaPublicKey

```
typedef struct otPlatCryptoEcdsaPublicKey otPlatCryptoEcdsaPublicKey
```

Definition at line 200 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaSignature

```
typedef struct otPlatCryptoEcdsaSignature otPlatCryptoEcdsaSignature
```

Definition at line 223 of file include/openthread/platform/crypto.h

Variable Documentation

OT_TOOL_PACKED_END

```
OT_TOOL_PACKED_BEGIN struct otPlatCryptoEcdsaSignature OT_TOOL_PACKED_END
```

Definition at line 152 of file include/openthread/platform/crypto.h

Function Documentation

otPlatCryptoInit

> void otPlatCryptoInit (void)

Initialize the Crypto module.

Parameters

N/A		

Definition at line `235` of file `include/openthread/platform/crypto.h`

otPlatCryptoImportKey

> otError otPlatCryptoImportKey (otCryptoKeyRef *aKeyRef, otCryptoKeyType aKeyType, otCryptoKeyAlgorithm aKeyAlgorithm, int aKeyUsage, otCryptoKeyStorage aKeyPersistence, const uint8_t *aKey, size_t aKeyLen)

Import a key into PSA ITS.

Parameters

[inout]	aKeyRef	Pointer to the key ref to be used for crypto operations.
[in]	aKeyType	Key Type encoding for the key.
[in]	aKeyAlgorithm	Key algorithm encoding for the key.
[in]	aKeyUsage	Key Usage encoding for the key (combinations of `OT_CRYPTO_KEY_USAGE_*`).
[in]	aKeyPersistence	Key Persistence for this key
[in]	aKey	Actual key to be imported.
[in]	aKeyLen	Length of the key to be imported.

Note

- If OT_CRYPTO_KEY_STORAGE_PERSISTENT is passed for aKeyPersistence then `aKeyRef` is input and platform should use the given aKeyRef and MUST not change it.

If OT_CRYPTO_KEY_STORAGE_VOLATILE is passed for aKeyPersistence then `aKeyRef` is output, the initial value does not matter and platform API MUST update it to return the new key ref.

This API is only used by OT core when `OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE` is enabled.

Definition at line `261` of file `include/openthread/platform/crypto.h`

otPlatCryptoExportKey

> otError otPlatCryptoExportKey (otCryptoKeyRef aKeyRef, uint8_t *aBuffer, size_t aBufferLen, size_t *aKeyLen)

Export a key stored in PSA ITS.

Parameters

[in]	aKeyRef	The key ref to be used for crypto operations.
[out]	aBuffer	Pointer to the buffer where key needs to be exported.
[in]	aBufferLen	Length of the buffer passed to store the exported key.

[out]	aKeyLen	Pointer to return the length of the exported key.

Note

- This API is only used by OT core when `OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE` is enabled.

Definition at line `284` of file `include/openthread/platform/crypto.h`

otPlatCryptoDestroyKey

```
otError otPlatCryptoDestroyKey (otCryptoKeyRef aKeyRef)
```

Destroy a key stored in PSA ITS.

Parameters

[in]	aKeyRef	The key ref to be destroyed

Note

- This API is only used by OT core when `OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE` is enabled.

Definition at line `297` of file `include/openthread/platform/crypto.h`

otPlatCryptoHasKey

```
bool otPlatCryptoHasKey (otCryptoKeyRef aKeyRef)
```

Check if the key ref passed has an associated key in PSA ITS.

Parameters

[in]	aKeyRef	The Key Ref to check.

Note

- This API is only used by OT core when `OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE` is enabled.

Definition at line `310` of file `include/openthread/platform/crypto.h`

otPlatCryptoHmacSha256Init

```
otError otPlatCryptoHmacSha256Init (otCryptoContext *aContext)
```

Initialize the HMAC operation.

Parameters

[in]	aContext	Context for HMAC operation.

Note

- The platform driver shall point the context to the correct object such as psa_mac_operation_t or mbedtls_md_context_t.

Definition at line `325` of file `include/openthread/platform/crypto.h`

otPlatCryptoHmacSha256Deinit

> otError otPlatCryptoHmacSha256Deinit (otCryptoContext *aContext)

Uninitialize the HMAC operation.

Parameters

[in]	aContext	Context for HMAC operation.

Definition at line `337` of file `include/openthread/platform/crypto.h`

otPlatCryptoHmacSha256Start

> otError otPlatCryptoHmacSha256Start (otCryptoContext *aContext, const otCryptoKey *aKey)

Start HMAC operation.

Parameters

[in]	aContext	Context for HMAC operation.
[in]	aKey	Key material to be used for HMAC operation.

Definition at line `350` of file `include/openthread/platform/crypto.h`

otPlatCryptoHmacSha256Update

> otError otPlatCryptoHmacSha256Update (otCryptoContext *aContext, const void *aBuf, uint16_t aBufLength)

Update the HMAC operation with new input.

Parameters

[in]	aContext	Context for HMAC operation.
[in]	aBuf	A pointer to the input buffer.
[in]	aBufLength	The length of `aBuf` in bytes.

Definition at line `364` of file `include/openthread/platform/crypto.h`

otPlatCryptoHmacSha256Finish

> otError otPlatCryptoHmacSha256Finish (otCryptoContext *aContext, uint8_t *aBuf, size_t aBufLength)

Complete the HMAC operation.

Parameters

[in]	aContext	Context for HMAC operation.
[out]	aBuf	A pointer to the output buffer.
[in]	aBufLength	The length of `aBuf` in bytes.

Definition at line `378` of file `include/openthread/platform/crypto.h`

otPlatCryptoAesInit

> otError otPlatCryptoAesInit (otCryptoContext *aContext)

Initialise the AES operation.

Parameters

[in]	aContext	Context for AES operation.

Note

- The platform driver shall point the context to the correct object such as psa_key_id or mbedtls_aes_context_t.

Definition at line `394` of file `include/openthread/platform/crypto.h`

otPlatCryptoAesSetKey

> otError otPlatCryptoAesSetKey (otCryptoContext *aContext, const otCryptoKey *aKey)

Set the key for AES operation.

Parameters

[in]	aContext	Context for AES operation.
[out]	aKey	Key to use for AES operation.

Definition at line `407` of file `include/openthread/platform/crypto.h`

otPlatCryptoAesEncrypt

> otError otPlatCryptoAesEncrypt (otCryptoContext *aContext, const uint8_t *aInput, uint8_t *aOutput)

Encrypt the given data.

Parameters

[in]	aContext	Context for AES operation.
[in]	aInput	Pointer to the input buffer.
[in]	aOutput	Pointer to the output buffer.

Definition at line `421` of file `include/openthread/platform/crypto.h`

otPlatCryptoAesFree

> otError otPlatCryptoAesFree (otCryptoContext *aContext)

Free the AES context.

Parameters

[in]	aContext	Context for AES operation.

Definition at line `433` of file `include/openthread/platform/crypto.h`

otPlatCryptoHkdfInit

otError otPlatCryptoHkdfInit (otCryptoContext *aContext)

Initialise the HKDF context.

Parameters

[in]	aContext	Context for HKDF operation.

Note

- The platform driver shall point the context to the correct object such as psa_key_derivation_operation_t or HmacSha256::Hash

Definition at line 448 of file include/openthread/platform/crypto.h

otPlatCryptoHkdfExpand

otError otPlatCryptoHkdfExpand (otCryptoContext *aContext, const uint8_t *aInfo, uint16_t aInfoLength, uint8_t *aOutputKey, uint16_t aOutputKeyLength)

Perform HKDF Expand step.

Parameters

[in]	aContext	Operation context for HKDF operation.
[in]	aInfo	Pointer to the Info sequence.
[in]	aInfoLength	Length of the Info sequence.
[out]	aOutputKey	Pointer to the output Key.
[in]	aOutputKeyLength	Size of the output key buffer.

Definition at line 464 of file include/openthread/platform/crypto.h

otPlatCryptoHkdfExtract

otError otPlatCryptoHkdfExtract (otCryptoContext *aContext, const uint8_t *aSalt, uint16_t aSaltLength, const otCryptoKey *aInputKey)

Perform HKDF Extract step.

Parameters

[in]	aContext	Operation context for HKDF operation.
[in]	aSalt	Pointer to the Salt for HKDF.
[in]	aSaltLength	Length of Salt.
[in]	aInputKey	Pointer to the input key.

Definition at line 482 of file include/openthread/platform/crypto.h

otPlatCryptoHkdfDeinit

otError otPlatCryptoHkdfDeinit (otCryptoContext *aContext)

Uninitialize the HKDF context.

Parameters

[in]	aContext	Context for HKDF operation.

Definition at line 497 of file include/openthread/platform/crypto.h

otPlatCryptoSha256Init

```
otError otPlatCryptoSha256Init (otCryptoContext *aContext)
```

Initialise the SHA-256 operation.

Parameters

[in]	aContext	Context for SHA-256 operation.

Note

- The platform driver shall point the context to the correct object such as psa_hash_operation_t or mbedtls_sha256_context.

Definition at line 512 of file include/openthread/platform/crypto.h

otPlatCryptoSha256Deinit

```
otError otPlatCryptoSha256Deinit (otCryptoContext *aContext)
```

Uninitialize the SHA-256 operation.

Parameters

[in]	aContext	Context for SHA-256 operation.

Definition at line 524 of file include/openthread/platform/crypto.h

otPlatCryptoSha256Start

```
otError otPlatCryptoSha256Start (otCryptoContext *aContext)
```

Start SHA-256 operation.

Parameters

[in]	aContext	Context for SHA-256 operation.

Definition at line 536 of file include/openthread/platform/crypto.h

otPlatCryptoSha256Update

```
otError otPlatCryptoSha256Update (otCryptoContext *aContext, const void *aBuf, uint16_t aBufLength)
```

Update SHA-256 operation with new input.

Parameters

[in]	aContext	Context for SHA-256 operation.
[in]	aBuf	A pointer to the input buffer.

[in]	aBufLength	The length of `aBuf` in bytes.

Definition at line `550` of file `include/openthread/platform/crypto.h`

otPlatCryptoSha256Finish

```
otError otPlatCryptoSha256Finish (otCryptoContext *aContext, uint8_t *aHash, uint16_t aHashSize)
```

Finish SHA-256 operation.

Parameters

[in]	aContext	Context for SHA-256 operation.
[in]	aHash	A pointer to the output buffer, where hash needs to be stored.
[in]	aHashSize	The length of `aHash` in bytes.

Definition at line `564` of file `include/openthread/platform/crypto.h`

otPlatCryptoRandomInit

```
void otPlatCryptoRandomInit (void)
```

Initialize cryptographically-secure pseudorandom number generator (CSPRNG).

Parameters

N/A		

Definition at line `570` of file `include/openthread/platform/crypto.h`

otPlatCryptoRandomDeinit

```
void otPlatCryptoRandomDeinit (void)
```

Deinitialize cryptographically-secure pseudorandom number generator (CSPRNG).

Parameters

N/A		

Definition at line `576` of file `include/openthread/platform/crypto.h`

otPlatCryptoRandomGet

```
otError otPlatCryptoRandomGet (uint8_t *aBuffer, uint16_t aSize)
```

Fills a given buffer with cryptographically secure random bytes.

Parameters

[out]	aBuffer	A pointer to a buffer to fill with the random bytes.
[in]	aSize	Size of buffer (number of bytes to fill).

Definition at line 588 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaGenerateKey

```
otError otPlatCryptoEcdsaGenerateKey (otPlatCryptoEcdsaKeyPair *aKeyPair)
```

Generate and populate the output buffer with a new ECDSA key-pair.

Parameters

[out]	aKeyPair	A pointer to an ECDSA key-pair structure to store the generated key-pair.

Definition at line 601 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaGetPublicKey

```
otError otPlatCryptoEcdsaGetPublicKey (const otPlatCryptoEcdsaKeyPair *aKeyPair, otPlatCryptoEcdsaPublicKey
*aPublicKey)
```

Get the associated public key from the input context.

Parameters

[in]	aKeyPair	A pointer to an ECDSA key-pair structure where the key-pair is stored.
[out]	aPublicKey	A pointer to an ECDSA public key structure to store the public key.

Definition at line 614 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaSign

```
otError otPlatCryptoEcdsaSign (const otPlatCryptoEcdsaKeyPair *aKeyPair, const otPlatCryptoSha256Hash *aHash,
otPlatCryptoEcdsaSignature *aSignature)
```

Calculate the ECDSA signature for a hashed message using the private key from the input context.

Parameters

[in]	aKeyPair	A pointer to an ECDSA key-pair structure where the key-pair is stored.
[in]	aHash	A pointer to a SHA-256 hash structure where the hash value for signature calculation is stored.
[out]	aSignature	A pointer to an ECDSA signature structure to output the calculated signature.

Uses the deterministic digital signature generation procedure from RFC 6979.

Definition at line 632 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaVerify

```
otError otPlatCryptoEcdsaVerify (const otPlatCryptoEcdsaPublicKey *aPublicKey, const otPlatCryptoSha256Hash *aHash,
const otPlatCryptoEcdsaSignature *aSignature)
```

Use the key from the input context to verify the ECDSA signature of a hashed message.

Parameters

[in]	aPublicKey	A pointer to an ECDSA public key structure where the public key for signature verification is stored.

[in]	aHash	A pointer to a SHA-256 hash structure where the hash value for signature verification is stored.
[in]	aSignature	A pointer to an ECDSA signature structure where the signature value to be verified is stored.

Definition at line 652 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaSignUsingKeyRef

> otError otPlatCryptoEcdsaSignUsingKeyRef (otCryptoKeyRef aKeyRef, const otPlatCryptoSha256Hash *aHash, otPlatCryptoEcdsaSignature *aSignature)

Calculate the ECDSA signature for a hashed message using the Key reference passed.

Parameters

[in]	aKeyRef	Key Reference to the slot where the key-pair is stored.
[in]	aHash	A pointer to a SHA-256 hash structure where the hash value for signature calculation is stored.
[out]	aSignature	A pointer to an ECDSA signature structure to output the calculated signature.

Uses the deterministic digital signature generation procedure from RFC 6979.

Note

- This API is only used by OT core when OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE is enabled.

Definition at line 674 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaExportPublicKey

> otError otPlatCryptoEcdsaExportPublicKey (otCryptoKeyRef aKeyRef, otPlatCryptoEcdsaPublicKey *aPublicKey)

Get the associated public key from the key reference passed.

Parameters

[in]	aKeyRef	Key Reference to the slot where the key-pair is stored.
[out]	aPublicKey	A pointer to an ECDSA public key structure to store the public key.

The public key is stored differently depending on the crypto backend library being used (OPENTHREAD_CONFIG_CRYPTO_LIB).

This API must make sure to return the public key as a byte sequence representation of an uncompressed curve point (RFC 6605 - sec 4)

Note

- This API is only used by OT core when OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE is enabled.

Definition at line 697 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaGenerateAndImportKey

> otError otPlatCryptoEcdsaGenerateAndImportKey (otCryptoKeyRef aKeyRef)

Generate and import a new ECDSA key-pair at reference passed.

Parameters

[in]	aKeyRef	Key Reference to the slot where the key-pair is stored.

Note

- This API is only used by OT core when `OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE` is enabled.

Definition at line `712` of file `include/openthread/platform/crypto.h`

otPlatCryptoEcdsaVerifyUsingKeyRef

```
otError otPlatCryptoEcdsaVerifyUsingKeyRef (otCryptoKeyRef aKeyRef, const otPlatCryptoSha256Hash *aHash, const otPlatCryptoEcdsaSignature *aSignature)
```

Use the keyref to verify the ECDSA signature of a hashed message.

Parameters

[in]	aKeyRef	Key Reference to the slot where the key-pair is stored.
[in]	aHash	A pointer to a SHA-256 hash structure where the hash value for signature verification is stored.
[in]	aSignature	A pointer to an ECDSA signature structure where the signature value to be verified is stored.

Note

- This API is only used by OT core when `OPENTHREAD_CONFIG_PLATFORM_KEY_REFERENCES_ENABLE` is enabled.

Definition at line `731` of file `include/openthread/platform/crypto.h`

otPlatCryptoPbkdf2GenerateKey

```
void otPlatCryptoPbkdf2GenerateKey (const uint8_t *aPassword, uint16_t aPasswordLen, const uint8_t *aSalt, uint16_t aSaltLen, uint32_t alterationCounter, uint16_t aKeyLen, uint8_t *aKey)
```

Perform PKCS#5 PBKDF2 using CMAC (AES-CMAC-PRF-128).

Parameters

[in]	aPassword	Password to use when generating key.
[in]	aPasswordLen	Length of password.
[in]	aSalt	Salt to use when generating key.
[in]	aSaltLen	Length of salt.
[in]	alterationCounter	Iteration count.
[in]	aKeyLen	Length of generated key in bytes.
[out]	aKey	A pointer to the generated key.

Definition at line `747` of file `include/openthread/platform/crypto.h`

Macro Definition Documentation

OT_CRYPTO_SHA256_HASH_SIZE

```
#define OT_CRYPTO_SHA256_HASH_SIZE
```

Value:

```
32
```

Length of SHA256 hash (in bytes).

Definition at line 140 of file include/openthread/platform/crypto.h

OT_CRYPTO_ECDSA_MAX_DER_SIZE

```
#define OT_CRYPTO_ECDSA_MAX_DER_SIZE
```

Value:

```
125
```

Max buffer size (in bytes) for representing the EDCSA key-pair in DER format.

Definition at line 164 of file include/openthread/platform/crypto.h

OT_CRYPTO_ECDSA_PUBLIC_KEY_SIZE

```
#define OT_CRYPTO_ECDSA_PUBLIC_KEY_SIZE
```

Value:

```
64
```

Buffer size (in bytes) for representing the EDCSA public key.

Definition at line 184 of file include/openthread/platform/crypto.h

OT_CRYPTO_ECDSA_SIGNATURE_SIZE

```
#define OT_CRYPTO_ECDSA_SIGNATURE_SIZE
```

Value:

```
64
```

Buffer size (in bytes) for representing the EDCSA signature.

Definition at line 206 of file include/openthread/platform/crypto.h

OT_CRYPTO_PBDKF2_MAX_SALT_SIZE

```
#define OT_CRYPTO_PBDKF2_MAX_SALT_SIZE
```

Value:

```
30
```

Max PBKDF2 SALT length: salt prefix (6) + extended panid (8) + network name (16)

Definition at line 229 of file include/openthread/platform/crypto.h

otCryptoKey

Represents the Key Material required for Crypto operations.

Public Attributes

const uint8_t *	mKey Pointer to the buffer containing key. NULL indicates to use `mKeyRef`.
uint16_t	mKeyLength The key length in bytes (applicable when `mKey` is not NULL).
uint32_t	mKeyRef The PSA key ref (requires `mKey` to be NULL).

Public Attribute Documentation

mKey

```
const uint8_t* otCryptoKey::mKey
```

Pointer to the buffer containing key. NULL indicates to use `mKeyRef`.

Definition at line `119` of file `include/openthread/platform/crypto.h`

mKeyLength

```
uint16_t otCryptoKey::mKeyLength
```

The key length in bytes (applicable when `mKey` is not NULL).

Definition at line `120` of file `include/openthread/platform/crypto.h`

mKeyRef

```
uint32_t otCryptoKey::mKeyRef
```

The PSA key ref (requires `mKey` to be NULL).

Definition at line `121` of file `include/openthread/platform/crypto.h`

otCryptoContext

Stores the context object for platform APIs.

Public Attributes

void *	mContext Pointer to the context.
uint16_t	mContextSize The length of the context in bytes.

Public Attribute Documentation

mContext

```
void* otCryptoContext::mContext
```

Pointer to the context.

Definition at line 132 of file include/openthread/platform/crypto.h

mContextSize

```
uint16_t otCryptoContext::mContextSize
```

The length of the context in bytes.

Definition at line 133 of file include/openthread/platform/crypto.h

otPlatCryptoSha256Hash

Represents a SHA-256 hash.

Public Attributes

uint8_t	m8 Hash bytes.

Public Attribute Documentation

m8

```
uint8_t otPlatCryptoSha256Hash::m8[OT_CRYPTO_SHA256_HASH_SIZE]
```

Hash bytes.

Definition at line 151 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaKeyPair

Represents an ECDSA key pair (public and private keys).

The key pair is stored using Distinguished Encoding Rules (DER) format (per RFC 5915).

Public Attributes

uint8_t	mDerBytes
uint8_t	mDerLength

Public Attribute Documentation

mDerBytes

```
uint8_t otPlatCryptoEcdsaKeyPair::mDerBytes[OT_CRYPTO_ECDSA_MAX_DER_SIZE]
```

Definition at line 176 of file include/openthread/platform/crypto.h

mDerLength

```
uint8_t otPlatCryptoEcdsaKeyPair::mDerLength
```

Definition at line 177 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaPublicKey

Represents a ECDSA public key.

The public key is stored as a byte sequence representation of an uncompressed curve point (RFC 6605 - sec 4).

Public Attributes

uint8_t m8

Public Attribute Documentation

m8

```
uint8_t otPlatCryptoEcdsaPublicKey::m8[OT_CRYPTO_ECDSA_PUBLIC_KEY_SIZE]
```

Definition at line 197 of file include/openthread/platform/crypto.h

otPlatCryptoEcdsaSignature

Represents an ECDSA signature.

The signature is encoded as the concatenated binary representation of two MPIs `r` and `s` which are calculated during signing (RFC 6605 - section 4).

Public Attributes

uint8_t	m8

Public Attribute Documentation

m8

```
uint8_t otPlatCryptoEcdsaSignature::m8[OT_CRYPTO_ECDSA_SIGNATURE_SIZE]
```

Definition at line `220` of file `include/openthread/platform/crypto.h`

DNS - Platform

This module includes the platform abstraction for sending recursive DNS query to upstream DNS servers.

Typedefs

typedef struct otPlatDnsUpstreamQuery	otPlatDnsUpstreamQuery This opaque type represents an upstream DNS query transaction.

Functions

void	otPlatDnsStartUpstreamQuery(otInstance *aInstance, otPlatDnsUpstreamQuery *aTxn, const otMessage *aQuery) Starts an upstream query transaction.
void	otPlatDnsCancelUpstreamQuery(otInstance *aInstance, otPlatDnsUpstreamQuery *aTxn) Cancels a transaction of upstream query.
void	otPlatDnsUpstreamQueryDone(otInstance *aInstance, otPlatDnsUpstreamQuery *aTxn, otMessage *aResponse) The platform calls this function to finish DNS query.

Typedef Documentation

otPlatDnsUpstreamQuery

```
typedef struct otPlatDnsUpstreamQuery otPlatDnsUpstreamQuery
```

This opaque type represents an upstream DNS query transaction.

Definition at line 60 of file include/openthread/platform/dns.h

Function Documentation

otPlatDnsStartUpstreamQuery

```
void otPlatDnsStartUpstreamQuery (otInstance *aInstance, otPlatDnsUpstreamQuery *aTxn, const otMessage *aQuery)
```

Starts an upstream query transaction.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aTxn	A pointer to the opaque DNS query transaction object.
[in]	aQuery	A message buffer of the DNS payload that should be sent to upstream DNS server.

In success case (and errors represented by DNS protocol messages), the platform is expected to call `otPlatDnsUpstreamQueryDone`.

- The OpenThread core may cancel a (possibly timeout) query transaction by calling `otPlatDnsCancelUpstreamQuery`, the platform must not call `otPlatDnsUpstreamQueryDone` on a cancelled transaction.

Definition at line `76` of file `include/openthread/platform/dns.h`

otPlatDnsCancelUpstreamQuery

> void otPlatDnsCancelUpstreamQuery (otInstance *aInstance, otPlatDnsUpstreamQuery *aTxn)

Cancels a transaction of upstream query.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aTxn	A pointer to the opaque DNS query transaction object.

The platform must call `otPlatDnsUpstreamQueryDone` to release the resources.

Definition at line `87` of file `include/openthread/platform/dns.h`

otPlatDnsUpstreamQueryDone

> void otPlatDnsUpstreamQueryDone (otInstance *aInstance, otPlatDnsUpstreamQuery *aTxn, otMessage *aResponse)

The platform calls this function to finish DNS query.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aTxn	A pointer to the opaque DNS query transaction object.
[in]	aResponse	A message buffer of the DNS response payload or `nullptr` to close a transaction without a response.

The transaction will be released, so the platform must not call on the same transaction twice. This function passes the ownership of `aResponse` to OpenThread stack.

Platform can pass a nullptr to close a transaction without a response.

Definition at line `103` of file `include/openthread/platform/dns.h`

Entropy

Entropy

This module includes the platform abstraction for entropy generation.

Functions

otError otPlatEntropyGet(uint8_t *aOutput, uint16_t aOutputLength)
Fill buffer with entropy.

Function Documentation

otPlatEntropyGet

> otError otPlatEntropyGet (uint8_t *aOutput, uint16_t aOutputLength)

Fill buffer with entropy.

Parameters

[out]	aOutput	A pointer to where the true random values are placed. Must not be NULL.
[in]	aOutputLength	Size of `aBuffer`.

MUST be implemented using a true random number generator (TRNG).

Definition at line `69` of file `include/openthread/platform/entropy.h`

Factory Diagnostics - Platform

Factory Diagnostics - Platform

This module includes the platform abstraction for diagnostics features.

Enumerations

enum otGpioMode {

OT_GPIO_MODE_INPUT = 0
OT_GPIO_MODE_OUTPUT = 1

}

Defines the gpio modes.

Functions

otError otPlatDiagProcess(otInstance *aInstance, uint8_t aArgsLength, char *aArgs[], char *aOutput, size_t aOutputMaxLen)

Processes a factory diagnostics command line.

void otPlatDiagModeSet(bool aMode)

Enables/disables the factory diagnostics mode.

bool otPlatDiagModeGet(void)

Indicates whether or not factory diagnostics mode is enabled.

void otPlatDiagChannelSet(uint8_t aChannel)

Sets the channel to use for factory diagnostics.

void otPlatDiagTxPowerSet(int8_t aTxPower)

Sets the transmit power to use for factory diagnostics.

void otPlatDiagRadioReceived(otInstance *aInstance, otRadioFrame *aFrame, otError aError)

Processes the received radio frame.

void otPlatDiagAlarmCallback(otInstance *aInstance)

Processes the alarm event.

otError otPlatDiagGpioSet(uint32_t aGpio, bool aValue)

Sets the gpio value.

otError otPlatDiagGpioGet(uint32_t aGpio, bool *aValue)

Gets the gpio value.

otError otPlatDiagGpioSetMode(uint32_t aGpio, otGpioMode aMode)

Sets the gpio mode.

otError otPlatDiagGpioGetMode(uint32_t aGpio, otGpioMode *aMode)

Gets the gpio mode.

otError otPlatDiagRadioSetRawPowerSetting(otInstance *aInstance, const uint8_t *aRawPowerSetting, uint16_t aRawPowerSettingLength)

Set the radio raw power setting for diagnostics module.

otError otPlatDiagRadioGetRawPowerSetting(otInstance *aInstance, uint8_t *aRawPowerSetting, uint16_t *aRawPowerSettingLength)
Get the radio raw power setting for diagnostics module.

otError otPlatDiagRadioRawPowerSettingEnable(otInstance *aInstance, bool aEnable)
Enable/disable the platform layer to use the raw power setting set by `otPlatDiagRadioSetRawPowerSetting()` .

otError otPlatDiagRadioTransmitCarrier(otInstance *aInstance, bool aEnable)
Start/stop the platform layer to transmit continuous carrier wave.

otError otPlatDiagRadioTransmitStream(otInstance *aInstance, bool aEnable)
Start/stop the platform layer to transmit stream of characters.

otError otPlatDiagRadioGetPowerSettings(otInstance *aInstance, uint8_t aChannel, int16_t *aTargetPower, int16_t *aActualPower, uint8_t *aRawPowerSetting, uint16_t *aRawPowerSettingLength)
Get the power settings for the given channel.

Enumeration Documentation

otGpioMode

> otGpioMode

Defines the gpio modes.

	Enumerator
OT_GPIO_MODE_INPUT	Input mode without pull resistor.
OT_GPIO_MODE_OUTPUT	Output mode.

Definition at line `63` of file `include/openthread/platform/diag.h`

Function Documentation

otPlatDiagProcess

> otError otPlatDiagProcess (otInstance *aInstance, uint8_t aArgsLength, char *aArgs[], char *aOutput, size_t aOutputMaxLen)

Processes a factory diagnostics command line.

Parameters

[in]	aInstance	The OpenThread instance for current request.
[in]	aArgsLength	The number of arguments in `aArgs` .
[in]	aArgs	The arguments of diagnostics command line.
[out]	aOutput	The diagnostics execution result.
[in]	aOutputMaxLen	The output buffer size.

The output of this function (the content written to `aOutput`) MUST terminate with `\0` and the `\0` is within the output buffer.

Definition at line `86` of file `include/openthread/platform/diag.h`

otPlatDiagModeSet

```
void otPlatDiagModeSet (bool aMode)
```

Enables/disables the factory diagnostics mode.

Parameters

[in]	aMode	TRUE to enable diagnostics mode, FALSE otherwise.

Definition at line 98 of file include/openthread/platform/diag.h

otPlatDiagModeGet

```
bool otPlatDiagModeGet (void)
```

Indicates whether or not factory diagnostics mode is enabled.

Parameters

N/A		

Returns

- TRUE if factory diagnostics mode is enabled, FALSE otherwise.

Definition at line 106 of file include/openthread/platform/diag.h

otPlatDiagChannelSet

```
void otPlatDiagChannelSet (uint8_t aChannel)
```

Sets the channel to use for factory diagnostics.

Parameters

[in]	aChannel	The channel value.

Definition at line 114 of file include/openthread/platform/diag.h

otPlatDiagTxPowerSet

```
void otPlatDiagTxPowerSet (int8_t aTxPower)
```

Sets the transmit power to use for factory diagnostics.

Parameters

[in]	aTxPower	The transmit power value.

Definition at line 122 of file include/openthread/platform/diag.h

otPlatDiagRadioReceived

```
void otPlatDiagRadioReceived (otInstance *aInstance, otRadioFrame *aFrame, otError aError)
```

Processes the received radio frame.

Parameters

[in]	aInstance	The OpenThread instance for current request.
[in]	aFrame	The received radio frame.
[in]	aError	The received radio frame status.

Definition at line 132 of file include/openthread/platform/diag.h

otPlatDiagAlarmCallback

```
void otPlatDiagAlarmCallback (otInstance *aInstance)
```

Processes the alarm event.

Parameters

[in]	aInstance	The OpenThread instance for current request.

Definition at line 140 of file include/openthread/platform/diag.h

otPlatDiagGpioSet

```
otError otPlatDiagGpioSet (uint32_t aGpio, bool aValue)
```

Sets the gpio value.

Parameters

[in]	aGpio	The gpio number.
[in]	aValue	true to set the gpio to high level, or false otherwise.

Definition at line 155 of file include/openthread/platform/diag.h

otPlatDiagGpioGet

```
otError otPlatDiagGpioGet (uint32_t aGpio, bool *aValue)
```

Gets the gpio value.

Parameters

[in]	aGpio	The gpio number.
[out]	aValue	A pointer where to put gpio value.

Definition at line 170 of file include/openthread/platform/diag.h

otPlatDiagGpioSetMode

```
otError otPlatDiagGpioSetMode (uint32_t aGpio, otGpioMode aMode)
```

Sets the gpio mode.

Parameters

[in]	aGpio	The gpio number.
[out]	aMode	The gpio mode.

Definition at line 185 of file include/openthread/platform/diag.h

otPlatDiagGpioGetMode

> otError otPlatDiagGpioGetMode (uint32_t aGpio, otGpioMode *aMode)

Gets the gpio mode.

Parameters

[in]	aGpio	The gpio number.
[out]	aMode	A pointer where to put gpio mode.

Definition at line 201 of file include/openthread/platform/diag.h

otPlatDiagRadioSetRawPowerSetting

> otError otPlatDiagRadioSetRawPowerSetting (otInstance *aInstance, const uint8_t *aRawPowerSetting, uint16_t aRawPowerSettingLength)

Set the radio raw power setting for diagnostics module.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aRawPowerSetting	A pointer to the raw power setting byte array.
[in]	aRawPowerSettingLength	The length of the `aRawPowerSetting`.

Definition at line 215 of file include/openthread/platform/diag.h

otPlatDiagRadioGetRawPowerSetting

> otError otPlatDiagRadioGetRawPowerSetting (otInstance *aInstance, uint8_t *aRawPowerSetting, uint16_t *aRawPowerSettingLength)

Get the radio raw power setting for diagnostics module.

Parameters

[in]	aInstance	The OpenThread instance structure.
[out]	aRawPowerSetting	A pointer to the raw power setting byte array.
[inout]	aRawPowerSettingLength	On input, a pointer to the size of `aRawPowerSetting`. On output, a pointer to the length of the raw power setting data.

Definition at line 234 of file include/openthread/platform/diag.h

otPlatDiagRadioRawPowerSettingEnable

otError otPlatDiagRadioRawPowerSettingEnable (otInstance *aInstance, bool aEnable)

Enable/disable the platform layer to use the raw power setting set by `otPlatDiagRadioSetRawPowerSetting()` .

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aEnable	TRUE to enable or FALSE to disable the raw power setting.

Definition at line `248` of file `include/openthread/platform/diag.h`

otPlatDiagRadioTransmitCarrier

otError otPlatDiagRadioTransmitCarrier (otInstance *aInstance, bool aEnable)

Start/stop the platform layer to transmit continuous carrier wave.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aEnable	TRUE to enable or FALSE to disable the platform layer to transmit continuous carrier wave.

Definition at line `261` of file `include/openthread/platform/diag.h`

otPlatDiagRadioTransmitStream

otError otPlatDiagRadioTransmitStream (otInstance *aInstance, bool aEnable)

Start/stop the platform layer to transmit stream of characters.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aEnable	TRUE to enable or FALSE to disable the platform layer to transmit stream.

Definition at line `274` of file `include/openthread/platform/diag.h`

otPlatDiagRadioGetPowerSettings

otError otPlatDiagRadioGetPowerSettings (otInstance *aInstance, uint8_t aChannel, int16_t *aTargetPower, int16_t *aActualPower, uint8_t *aRawPowerSetting, uint16_t *aRawPowerSettingLength)

Get the power settings for the given channel.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aChannel	The radio channel.
[out]	aTargetPower	The target power in 0.01 dBm.
[out]	aActualPower	The actual power in 0.01 dBm.
[out]	aRawPowerSetting	A pointer to the raw power setting byte array.

[inout]	aRawPowerSettingLength	On input, a pointer to the size of `aRawPowerSetting`. On output, a pointer to the length of the raw power setting data.

Definition at line `294` of file `include/openthread/platform/diag.h`

Logging - Platform

Logging - Platform

This module includes the platform abstraction for the debug log service.

Enumerations

enum otLogRegion {

OT_LOG_REGION_API = 1
OT_LOG_REGION_MLE = 2
OT_LOG_REGION_ARP = 3
OT_LOG_REGION_NET_DATA = 4
OT_LOG_REGION_ICMP = 5
OT_LOG_REGION_IP6 = 6
OT_LOG_REGION_TCP = 7
OT_LOG_REGION_MAC = 8
OT_LOG_REGION_MEM = 9
OT_LOG_REGION_NCP = 10
OT_LOG_REGION_MESH_COP = 11
OT_LOG_REGION_NET_DIAG = 12
OT_LOG_REGION_PLATFORM = 13
OT_LOG_REGION_COAP = 14
OT_LOG_REGION_CLI = 15
OT_LOG_REGION_CORE = 16
OT_LOG_REGION_UTIL = 17
OT_LOG_REGION_BBR = 18
OT_LOG_REGION_MLR = 19
OT_LOG_REGION_DUA = 20
OT_LOG_REGION_BR = 21
OT_LOG_REGION_SRP = 22
OT_LOG_REGION_DNS = 23

}

Represents log regions.

Typedefs

typedef int otLogLevel

Represents the log level.

typedef enum otLogRegion otLogRegion

Represents log regions.

Functions

void otPlatLog(otLogLevel aLogLevel, otLogRegion aLogRegion, const char *aFormat,...)

Outputs logs.

void otPlatLogHandleLevelChanged(otLogLevel aLogLevel)

Handles OpenThread log level changes.

Macros

#define OT_LOG_LEVEL_NONE 0
Log level None.

#define OT_LOG_LEVEL_CRIT 1
Log level Critical.

#define OT_LOG_LEVEL_WARN 2
Log level Warning.

#define OT_LOG_LEVEL_NOTE 3
Log level Notice.

#define OT_LOG_LEVEL_INFO 4
Log level Informational.

#define OT_LOG_LEVEL_DEBG 5
Log level Debug.

Enumeration Documentation

otLogRegion

> otLogRegion

Represents log regions.

The support for log region is removed and instead each core module can define its own name to appended to the logs. However, the `otLogRegion` enumeration is still defined as before to help with platforms which we may be using it in their `otPlatLog()` implementation. The OT core will always emit all logs with `OT_LOG_REGION_CORE`.

Enumerator	
OT_LOG_REGION_API	OpenThread API.
OT_LOG_REGION_MLE	MLE.
OT_LOG_REGION_ARP	EID-to-RLOC mapping.
OT_LOG_REGION_NET_DATA	Network Data.
OT_LOG_REGION_ICMP	ICMPv6.
OT_LOG_REGION_IP6	IPv6.
OT_LOG_REGION_TCP	TCP.
OT_LOG_REGION_MAC	IEEE 802.15.4 MAC.
OT_LOG_REGION_MEM	Memory.
OT_LOG_REGION_NCP	NCP.
OT_LOG_REGION_MESH_COP	Mesh Commissioning Protocol.
OT_LOG_REGION_NET_DIAG	Network Diagnostic.
OT_LOG_REGION_PLATFORM	Platform.
OT_LOG_REGION_COAP	CoAP.
OT_LOG_REGION_CLI	CLI.
OT_LOG_REGION_CORE	OpenThread Core.
OT_LOG_REGION_UTIL	Utility module.
OT_LOG_REGION_BBR	Backbone Router (available since Thread 1.2)
OT_LOG_REGION_MLR	Multicast Listener Registration (available since Thread 1.2)
OT_LOG_REGION_DUA	Domain Unicast Address (available since Thread 1.2)

OT_LOG_REGION_BR	Border Router.
OT_LOG_REGION_SRP	Service Registration Protocol (SRP)
OT_LOG_REGION_DNS	DNS.

Definition at line 123 of file include/openthread/platform/logging.h

Typedef Documentation

otLogLevel

```
typedef int otLogLevel
```

Represents the log level.

Definition at line 113 of file include/openthread/platform/logging.h

otLogRegion

```
typedef enum otLogRegion otLogRegion
```

Represents log regions.

The support for log region is removed and instead each core module can define its own name to appended to the logs. However, the `otLogRegion` enumeration is still defined as before to help with platforms which we may be using it in their `otPlatLog()` implementation. The OT core will always emit all logs with `OT_LOG_REGION_CORE`.

Definition at line 148 of file include/openthread/platform/logging.h

Function Documentation

otPlatLog

```
void otPlatLog (otLogLevel aLogLevel, otLogRegion aLogRegion, const char *aFormat,...)
```

Outputs logs.

Parameters

[in]	aLogLevel	The log level.
[in]	aLogRegion	The log region.
[in]	aFormat	A pointer to the format string.
[in]		Arguments for the format specification.

Note that the support for log region is removed. The OT core will always emit all logs with `OT_LOG_REGION_CORE` as `aLogRegion`.

Definition at line 162 of file include/openthread/platform/logging.h

otPlatLogHandleLevelChanged

```
void otPlatLogHandleLevelChanged (otLogLevel aLogLevel)
```

Handles OpenThread log level changes.

Parameters

[in]	aLogLevel	The new OpenThread log level.

This platform function is called whenever the OpenThread log level changes. This platform function is optional since an empty weak implementation has been provided.

Note

- Only applicable when `OPENTHREAD_CONFIG_LOG_LEVEL_DYNAMIC_ENABLE=1`.

Definition at line `175` of file `include/openthread/platform/logging.h`

Macro Definition Documentation

OT_LOG_LEVEL_NONE

```
#define OT_LOG_LEVEL_NONE
```

Value:

```
0
```

Log level None.

Note

- Log Levels are defines so that embedded implementations can eliminate code at compile time via #if/#else/#endif.

Definition at line `62` of file `include/openthread/platform/logging.h`

OT_LOG_LEVEL_CRIT

```
#define OT_LOG_LEVEL_CRIT
```

Value:

```
1
```

Log level Critical.

Note

- Log Levels are defines so that embedded implementations can eliminate code at compile time via #if/#else/#endif.

Definition at line `71` of file `include/openthread/platform/logging.h`

OT_LOG_LEVEL_WARN

```
#define OT_LOG_LEVEL_WARN
```

Value:

```
2
```

Log level Warning.

Note

- Log Levels are defines so that embedded implementations can eliminate code at compile time via #if/#else/#endif.

Definition at line 80 of file include/openthread/platform/logging.h

OT_LOG_LEVEL_NOTE

```
#define OT_LOG_LEVEL_NOTE
```

Value:

```
3
```

Log level Notice.

Note

- Log Levels are defines so that embedded implementations can eliminate code at compile time via #if/#else/#endif.

Definition at line 89 of file include/openthread/platform/logging.h

OT_LOG_LEVEL_INFO

```
#define OT_LOG_LEVEL_INFO
```

Value:

```
4
```

Log level Informational.

Note

- Log Levels are defines so that embedded implementations can eliminate code at compile time via #if/#else/#endif.

Definition at line 98 of file include/openthread/platform/logging.h

OT_LOG_LEVEL_DEBG

```
#define OT_LOG_LEVEL_DEBG
```

Value:

```
5
```

Log level Debug.

Note

- Log Levels are defines so that embedded implementations can eliminate code at compile time via #if/#else/#endif.

Definition at line 107 of file include/openthread/platform/logging.h

Memory

Memory

This module includes the platform abstraction for dynamic memory allocation.

Functions

void *	otPlatCAlloc(size_t aNum, size_t aSize) Dynamically allocates new memory.
void	otPlatFree(void *aPtr) Frees memory that was dynamically allocated.

Function Documentation

otPlatCAlloc

```
void * otPlatCAlloc (size_t aNum, size_t aSize)
```

Dynamically allocates new memory.

Parameters

[in]	aNum	The number of blocks to allocate
[in]	aSize	The size of each block to allocate

On platforms that support it, should just redirect to calloc. For those that don't support calloc, should support the same functionality:

"The calloc() function contiguously allocates enough space for count objects that are size bytes of memory each and returns a pointer to the allocated memory. The allocated memory is filled with bytes of value zero."

Is required for OPENTHREAD_CONFIG_HEAP_EXTERNAL_ENABLE.

Definition at line `74` of file `include/openthread/platform/memory.h`

otPlatFree

```
void otPlatFree (void *aPtr)
```

Frees memory that was dynamically allocated.

Parameters

[in]	aPtr	A pointer the memory blocks to free. The pointer may be NULL.

Is required for OPENTHREAD_CONFIG_HEAP_EXTERNAL_ENABLE.

Definition at line `83` of file `include/openthread/platform/memory.h`

Message Pool

Message Pool

This module includes the platform abstraction for the message pool.

Modules

otMessageBuffer

Typedefs

typedef struct otMessageBuffer	otMessageBuffer Represents an OpenThread message buffer.

Functions

void	otPlatMessagePoolInit(otInstance *aInstance, uint16_t aMinNumFreeBuffers, size_t aBufferSize) Initialize the platform implemented message pool.
otMessageBuffer *	otPlatMessagePoolNew(otInstance *aInstance) Allocate a buffer from the platform managed buffer pool.
void	otPlatMessagePoolFree(otInstance *aInstance, otMessageBuffer *aBuffer) Is used to free a buffer back to the platform managed buffer pool.
uint16_t	otPlatMessagePoolNumFreeBuffers(otInstance *aInstance) Get the number of free buffers.

Typedef Documentation

otMessageBuffer

```
typedef struct otMessageBuffer otMessageBuffer
```

Represents an OpenThread message buffer.

Definition at line 63 of file include/openthread/platform/messagepool.h

Function Documentation

otPlatMessagePoolInit

```
void otPlatMessagePoolInit (otInstance *aInstance, uint16_t aMinNumFreeBuffers, size_t aBufferSize)
```

Initialize the platform implemented message pool.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

[in]	aMinNumFreeBuffers	An uint16 containing the minimum number of free buffers desired by OpenThread.
[in]	aBufferSize	The size in bytes of a buffer object.

Is used when `OPENTHREAD_CONFIG_PLATFORM_MESSAGE_MANAGEMENT` is enabled.

Definition at line `75` of file `include/openthread/platform/messagepool.h`

otPlatMessagePoolNew

```
otMessageBuffer * otPlatMessagePoolNew (otInstance *aInstance)
```

Allocate a buffer from the platform managed buffer pool.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Is used when `OPENTHREAD_CONFIG_PLATFORM_MESSAGE_MANAGEMENT` is enabled.

The returned buffer instance MUST have at least `aBufferSize` bytes (as specified in `otPlatMessagePoolInit()`).

Returns

- A pointer to the buffer or NULL if no buffers are available.

Definition at line `89` of file `include/openthread/platform/messagepool.h`

otPlatMessagePoolFree

```
void otPlatMessagePoolFree (otInstance *aInstance, otMessageBuffer *aBuffer)
```

Is used to free a buffer back to the platform managed buffer pool.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.
[in]	aBuffer	The buffer to free.

Is used when `OPENTHREAD_CONFIG_PLATFORM_MESSAGE_MANAGEMENT` is enabled.

Definition at line `100` of file `include/openthread/platform/messagepool.h`

otPlatMessagePoolNumFreeBuffers

```
uint16_t otPlatMessagePoolNumFreeBuffers (otInstance *aInstance)
```

Get the number of free buffers.

Parameters

[in]	aInstance	A pointer to the OpenThread instance.

Is used when `OPENTHREAD_CONFIG_PLATFORM_MESSAGE_MANAGEMENT` is enabled.

Returns

- The number of buffers currently free and available to OpenThread.

Definition at line 112 of file include/openthread/platform/messagepool.h

otMessageBuffer

Represents an OpenThread message buffer.

Public Attributes

struct otMessageBuffer *	mNext Pointer to the next buffer.

Public Attribute Documentation

mNext

```
struct otMessageBuffer* otMessageBuffer::mNext
```

Pointer to the next buffer.

Definition at line 62 of file include/openthread/platform/messagepool.h

Miscellaneous

Miscellaneous

This module includes platform abstractions for miscellaneous behaviors.

Enumerations

enum otPlatResetReason {

OT_PLAT_RESET_REASON_POWER_ON = 0
OT_PLAT_RESET_REASON_EXTERNAL = 1
OT_PLAT_RESET_REASON_SOFTWARE = 2
OT_PLAT_RESET_REASON_FAULT = 3
OT_PLAT_RESET_REASON_CRASH = 4
OT_PLAT_RESET_REASON_ASSERT = 5
OT_PLAT_RESET_REASON_OTHER = 6
OT_PLAT_RESET_REASON_UNKNOWN = 7
OT_PLAT_RESET_REASON_WATCHDOG = 8
OT_PLAT_RESET_REASON_COUNT

}

Enumeration of possible reset reason codes.

enum otPlatMcuPowerState {

OT_PLAT_MCU_POWER_STATE_ON = 0
OT_PLAT_MCU_POWER_STATE_LOW_POWER = 1
OT_PLAT_MCU_POWER_STATE_OFF = 2

}

Enumeration of micro-controller's power states.

Functions

void otPlatReset(otInstance *aInstance)

Performs a software reset on the platform, if supported.

otError otPlatResetToBootloader(otInstance *aInstance)

Performs a hardware reset on the platform to launch bootloader mode, if supported.

otPlatResetReason otPlatGetResetReason(otInstance *aInstance)

Returns the reason for the last platform reset.

void otPlatAssertFail(const char *aFilename, int aLineNumber)

Provides a platform specific implementation for assert.

void otPlatWakeHost(void)

Performs a platform specific operation to wake the host MCU.

otError otPlatSetMcuPowerState(otInstance *aInstance, otPlatMcuPowerState aState)

Sets the desired MCU power state.

otPlatMcuPowerState otPlatGetMcuPowerState(otInstance *aInstance)

Gets the current desired MCU power state.

Enumeration Documentation

otPlatResetReason

> otPlatResetReason

Enumeration of possible reset reason codes.

These are in the same order as the Spinel reset reason codes.

Enumerator	
OT_PLAT_RESET_REASON_POWER_ON	
OT_PLAT_RESET_REASON_EXTERNAL	
OT_PLAT_RESET_REASON_SOFTWARE	
OT_PLAT_RESET_REASON_FAULT	
OT_PLAT_RESET_REASON_CRASH	
OT_PLAT_RESET_REASON_ASSERT	
OT_PLAT_RESET_REASON_OTHER	
OT_PLAT_RESET_REASON_UNKNOWN	
OT_PLAT_RESET_REASON_WATCHDOG	
OT_PLAT_RESET_REASON_COUNT	

Definition at line `84` of file `include/openthread/platform/misc.h`

otPlatMcuPowerState

> otPlatMcuPowerState

Enumeration of micro-controller's power states.

These values are used for NCP configuration when `OPENTHREAD_CONFIG_NCP_ENABLE_MCU_POWER_STATE_CONTROL` is enabled.

The power state specifies the desired power state of NCP's micro-controller (MCU) when the underlying platform's operating system enters idle mode (i.e., all active tasks/events are processed and the MCU can potentially enter a energy-saving power state).

The power state primarily determines how the host should interact with the NCP and whether the host needs an external trigger (a "poke") to NCP before it can communicate with the NCP or not.

After a reset, the MCU power state MUST be `OT_PLAT_POWER_STATE_ON`.

Enumerator	
OT_PLAT_MCU_POWER_STATE_ON	NCP's MCU stays on and active all the time.
OT_PLAT_MCU_POWER_STATE_LOW_POWER	NCP's MCU can enter low-power (energy-saving) state.
OT_PLAT_MCU_POWER_STATE_OFF	NCP is fully off.

Definition at line `138` of file `include/openthread/platform/misc.h`

Function Documentation

otPlatReset

> void otPlatReset (otInstance *aInstance)

Performs a software reset on the platform, if supported.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `62` of file `include/openthread/platform/misc.h`

otPlatResetToBootloader

```
otError otPlatResetToBootloader (otInstance *aInstance)
```

Performs a hardware reset on the platform to launch bootloader mode, if supported.

Parameters

[in]	aInstance	The OpenThread instance structure.

Used when `OPENTHREAD_CONFIG_PLATFORM_BOOTLOADER_MODE_ENABLE` is enabled.

Definition at line `76` of file `include/openthread/platform/misc.h`

otPlatGetResetReason

```
otPlatResetReason otPlatGetResetReason (otInstance *aInstance)
```

Returns the reason for the last platform reset.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `105` of file `include/openthread/platform/misc.h`

otPlatAssertFail

```
void otPlatAssertFail (const char *aFilename, int aLineNumber)
```

Provides a platform specific implementation for assert.

Parameters

[in]	aFilename	The name of the file where the assert occurred.
[in]	aLineNumber	The line number in the file where the assert occurred.

Definition at line `114` of file `include/openthread/platform/misc.h`

otPlatWakeHost

```
void otPlatWakeHost (void)
```

Performs a platform specific operation to wake the host MCU.

Parameters

N/A		

This is used only for NCP configurations.

Definition at line 121 of file include/openthread/platform/misc.h

otPlatSetMcuPowerState

```
otError otPlatSetMcuPowerState (otInstance *aInstance, otPlatMcuPowerState aState)
```

Sets the desired MCU power state.

Parameters

[in]	aInstance	A pointer to OpenThread instance.
[in]	aState	The new MCU power state.

This is only applicable and used for NCP configuration when OPENTHREAD_CONFIG_NCP_ENABLE_MCU_POWER_STATE_CONTROL is enabled.

Definition at line 192 of file include/openthread/platform/misc.h

otPlatGetMcuPowerState

```
otPlatMcuPowerState otPlatGetMcuPowerState (otInstance *aInstance)
```

Gets the current desired MCU power state.

Parameters

[in]	aInstance	A pointer to OpenThread instance.

This is only applicable and used for NCP configuration when OPENTHREAD_CONFIG_NCP_ENABLE_MCU_POWER_STATE_CONTROL is enabled.

After a reset, the power state MUST return OT_PLAT_POWER_STATE_ON. During operation, power state SHOULD only change through an explicit successful call to otPlatSetMcuPowerState().

Returns

- The current power state.

Definition at line 208 of file include/openthread/platform/misc.h

Network Simulator

Network Simulator

This module includes the platform abstraction for OTNS.

Functions

void otPlatOtnsStatus(const char *aStatus)
Exports status information to OTNS.

Function Documentation

otPlatOtnsStatus

> void otPlatOtnsStatus (const char *aStatus)

Exports status information to OTNS.

Parameters

[in]	aStatus	The status string.

The status information is represented by a null-terminated string with format recognizable by OTNS. Each call to `otPlatOtnsStatus` can send multiple statuses, separated by ';', e.x. "parid=577fbc37;lrid=5". Each status contains key and value separated by '='. Status value can be further separated into multiple fields using ',', e.x. "ping_request=fdde:ad00:beef:0:459e:d7b4:b65e:5480,4,112000".

New statuses should follow these conventions.

Currently, OTNS only supports virtual time simulation.

Definition at line `72` of file `include/openthread/platform/otns.h`

Radio

Radio

This module includes the platform abstraction for radio communication.

Modules

Radio Types

Radio Configuration

Radio Operation

Radio Extension

Radio Types

Radio Types

This module includes the platform abstraction for a radio frame.

Modules

otExtAddress

otMacKey

otMacKeyMaterial

otRadioIeInfo

otRadioFrame

otRadioCoexMetrics

otLinkMetrics

Enumerations

enum @12 {

OT_RADIO_FRAME_MAX_SIZE = 127
OT_RADIO_FRAME_MIN_SIZE = 3
OT_RADIO_SYMBOLS_PER_OCTET = 2
OT_RADIO_BIT_RATE = 250000
OT_RADIO_BITS_PER_OCTET = 8
OT_RADIO_SYMBOL_RATE = 62500
OT_RADIO_SYMBOL_TIME = 1000000 * 1 / OT_RADIO_SYMBOL_RATE
OT_RADIO_TEN_SYMBOLS_TIME = 10 * OT_RADIO_SYMBOL_TIME
OT_RADIO_LQI_NONE = 0
OT_RADIO_RSSI_INVALID = 127
OT_RADIO_POWER_INVALID = 127

}

enum @13 {

OT_RADIO_CHANNEL_PAGE_0 = 0
OT_RADIO_CHANNEL_PAGE_0_MASK = (1U << OT_RADIO_CHANNEL_PAGE_0)
OT_RADIO_CHANNEL_PAGE_2 = 2
OT_RADIO_CHANNEL_PAGE_2_MASK = (1U << OT_RADIO_CHANNEL_PAGE_2)

}

Defines the channel page.

enum @14 {

OT_RADIO_915MHZ_OQPSK_CHANNEL_MIN = 1
OT_RADIO_915MHZ_OQPSK_CHANNEL_MAX = 10
OT_RADIO_915MHZ_OQPSK_CHANNEL_MASK = 0x3ff << OT_RADIO_915MHZ_OQPSK_CHANNEL_MIN
OT_RADIO_2P4GHZ_OQPSK_CHANNEL_MIN = 11
OT_RADIO_2P4GHZ_OQPSK_CHANNEL_MAX = 26
OT_RADIO_2P4GHZ_OQPSK_CHANNEL_MASK = 0xffff << OT_RADIO_2P4GHZ_OQPSK_CHANNEL_MIN

}

Defines the frequency band channel range.

enum @15 {

OT_RADIO_CAPS_NONE = 0
OT_RADIO_CAPS_ACK_TIMEOUT = 1 << 0
OT_RADIO_CAPS_ENERGY_SCAN = 1 << 1
OT_RADIO_CAPS_TRANSMIT_RETRIES = 1 << 2
OT_RADIO_CAPS_CSMA_BACKOFF = 1 << 3
OT_RADIO_CAPS_SLEEP_TO_TX = 1 << 4
OT_RADIO_CAPS_TRANSMIT_SEC = 1 << 5
OT_RADIO_CAPS_TRANSMIT_TIMING = 1 << 6
OT_RADIO_CAPS_RECEIVE_TIMING = 1 << 7

}

Defines constants that are used to indicate different radio capabilities.

enum @16 {

OT_IE_HEADER_SIZE = 2
OT_CSL_IE_SIZE = 4
OT_ACK_IE_MAX_SIZE = 16
OT_ENH_PROBING_IE_DATA_MAX_SIZE = 2

}

Defines constants about size of header IE in ACK.

enum otRadioKeyType {

OT_KEY_TYPE_LITERAL_KEY = 0
OT_KEY_TYPE_KEY_REF = 1

}

Defines constants about key types.

enum otRadioState {

OT_RADIO_STATE_DISABLED = 0
OT_RADIO_STATE_SLEEP = 1
OT_RADIO_STATE_RECEIVE = 2
OT_RADIO_STATE_TRANSMIT = 3
OT_RADIO_STATE_INVALID = 255

}

Represents the state of a radio.

Typedefs

typedef uint8_t otRadioCaps

Represents radio capabilities.

typedef uint16_t otPanId

Represents the IEEE 802.15.4 PAN ID.

typedef uint16_t otShortAddress

Represents the IEEE 802.15.4 Short Address.

typedef struct otExtAddress otExtAddress

Represents the IEEE 802.15.4 Extended Address.

typedef struct otMacKey otMacKey

Represents a MAC Key.

typedef otCryptoKeyRef otMacKeyRef

Represents a MAC Key Ref used by PSA.

typedef struct otMacKeyMaterial otMacKeyMaterial

typedef struct otRadioIeInfo
Represents the IEEE 802.15.4 Header IE (Information Element) related information of a radio frame.

typedef struct otRadioFrame
Represents an IEEE 802.15.4 radio frame.

typedef enum otRadioState
Represents the state of a radio.

typedef struct otRadioCoexMetrics
The following are valid radio state transitions:

typedef struct otLinkMetrics
Represents what metrics are specified to query.

Variables

OT_TOOL_PACKED_BEGIN struct otExtAddress
OT_TOOL_PACKED_END

Macros

#define OT_PANID_BROADCAST 0xffff
IEEE 802.15.4 Broadcast PAN ID.

#define OT_EXT_ADDRESS_SIZE 8
Size of an IEEE 802.15.4 Extended Address (bytes)

#define CSL_IE_HEADER_BYTES_LO 0×04
Fixed CSL IE header first byte.

#define CSL_IE_HEADER_BYTES_HI 0×0d
Fixed CSL IE header second byte.

#define OT_MAC_KEY_SIZE 16
Size of the MAC Key in bytes.

#define OT_TOOL_PACKED_END undefined
Compiler-specific indication at the end of a byte packed class or struct.

Enumeration Documentation

@12

> @12

Enumerator	
OT_RADIO_FRAME_MAX_SIZE	aMaxPHYPacketSize (IEEE 802.15.4-2006)
OT_RADIO_FRAME_MIN_SIZE	Minimal size of frame FCS + CONTROL.
OT_RADIO_SYMBOLS_PER_OCTET	2.4 GHz IEEE 802.15.4-2006
OT_RADIO_BIT_RATE	2.4 GHz IEEE 802.15.4 (bits per second)
OT_RADIO_BITS_PER_OCTET	Number of bits per octet.
OT_RADIO_SYMBOL_RATE	The O-QPSK PHY symbol rate when operating in the 780MHz, 915MHz, 2380MHz, 2450MHz.

OT_RADIO_SYMBOL_TIME	Symbol duration time in unit of microseconds.
OT_RADIO_TEN_SYMBOLS_TIME	Time for 10 symbols in unit of microseconds.
OT_RADIO_LQI_NONE	LQI measurement not supported.
OT_RADIO_RSSI_INVALID	Invalid or unknown RSSI value.
OT_RADIO_POWER_INVALID	Invalid or unknown power value.

Definition at line `69` of file `include/openthread/platform/radio.h`

@13

> @13

Defines the channel page.

Enumerator	
OT_RADIO_CHANNEL_PAGE_0	2.4 GHz IEEE 802.15.4-2006
OT_RADIO_CHANNEL_PAGE_0_MASK	2.4 GHz IEEE 802.15.4-2006
OT_RADIO_CHANNEL_PAGE_2	915 MHz IEEE 802.15.4-2006
OT_RADIO_CHANNEL_PAGE_2_MASK	915 MHz IEEE 802.15.4-2006

Definition at line `94` of file `include/openthread/platform/radio.h`

@14

> @14

Defines the frequency band channel range.

Enumerator	
OT_RADIO_915MHZ_OQPSK_CHANNEL_MIN	915 MHz IEEE 802.15.4-2006
OT_RADIO_915MHZ_OQPSK_CHANNEL_MAX	915 MHz IEEE 802.15.4-2006
OT_RADIO_915MHZ_OQPSK_CHANNEL_MASK	915 MHz IEEE 802.15.4-2006
OT_RADIO_2P4GHZ_OQPSK_CHANNEL_MIN	2.4 GHz IEEE 802.15.4-2006
OT_RADIO_2P4GHZ_OQPSK_CHANNEL_MAX	2.4 GHz IEEE 802.15.4-2006
OT_RADIO_2P4GHZ_OQPSK_CHANNEL_MASK	2.4 GHz IEEE 802.15.4-2006

Definition at line `106` of file `include/openthread/platform/radio.h`

@15

> @15

Defines constants that are used to indicate different radio capabilities.

See `otRadioCaps`.

Enumerator	
OT_RADIO_CAPS_NONE	Radio supports no capability.
OT_RADIO_CAPS_ACK_TIMEOUT	Radio supports AckTime event.

OT_RADIO_CAPS_ENERGY_SCAN	Radio supports Energy Scans.
OT_RADIO_CAPS_TRANSMIT_RETRIES	Radio supports tx retry logic with collision avoidance (CSMA).
OT_RADIO_CAPS_CSMA_BACKOFF	Radio supports CSMA backoff for frame transmission (but no retry).
OT_RADIO_CAPS_SLEEP_TO_TX	Radio supports direct transition from sleep to TX with CSMA.
OT_RADIO_CAPS_TRANSMIT_SEC	Radio supports tx security.
OT_RADIO_CAPS_TRANSMIT_TIMING	Radio supports tx at specific time.
OT_RADIO_CAPS_RECEIVE_TIMING	Radio supports rx at specific time.

Definition at line 128 of file include/openthread/platform/radio.h

@16

> @16

Defines constants about size of header IE in ACK.

Enumerator	
OT_IE_HEADER_SIZE	Size of IE header in bytes.
OT_CSL_IE_SIZE	Size of CSL IE content in bytes.
OT_ACK_IE_MAX_SIZE	Max length for header IE in ACK.
OT_ENH_PROBING_IE_DATA_MAX_SIZE	Max length of Link Metrics data in Vendor-Specific IE.

Definition at line 161 of file include/openthread/platform/radio.h

otRadioKeyType

> otRadioKeyType

Defines constants about key types.

Enumerator	
OT_KEY_TYPE_LITERAL_KEY	Use Literal Keys.
OT_KEY_TYPE_KEY_REF	Use Reference to Key.

Definition at line 235 of file include/openthread/platform/radio.h

otRadioState

> otRadioState

Represents the state of a radio.

Initially, a radio is in the Disabled state.

Enumerator	
OT_RADIO_STATE_DISABLED	
OT_RADIO_STATE_SLEEP	
OT_RADIO_STATE_RECEIVE	
OT_RADIO_STATE_TRANSMIT	
OT_RADIO_STATE_INVALID	

Definition at line 389 of file include/openthread/platform/radio.h

Typedef Documentation

otRadioCaps

```
typedef uint8_t otRadioCaps
```

Represents radio capabilities.

The value is a bit-field indicating the capabilities supported by the radio. See OT_RADIO_CAPS_* definitions.

Definition at line 122 of file include/openthread/platform/radio.h

otPanId

```
typedef uint16_t otPanId
```

Represents the IEEE 802.15.4 PAN ID.

Definition at line 147 of file include/openthread/platform/radio.h

otShortAddress

```
typedef uint16_t otShortAddress
```

Represents the IEEE 802.15.4 Short Address.

Definition at line 153 of file include/openthread/platform/radio.h

otExtAddress

```
typedef struct otExtAddress otExtAddress
```

Represents the IEEE 802.15.4 Extended Address.

Definition at line 188 of file include/openthread/platform/radio.h

otMacKey

```
typedef struct otMacKey otMacKey
```

Represents a MAC Key.

Definition at line 208 of file include/openthread/platform/radio.h

otMacKeyRef

```
typedef otCryptoKeyRef otMacKeyRef
```

Represents a MAC Key Ref used by PSA.

Definition at line 214 of file include/openthread/platform/radio.h

otMacKeyMaterial

```
typedef struct otMacKeyMaterial otMacKeyMaterial
```

Definition at line 229 of file include/openthread/platform/radio.h

otRadioIeInfo

```
typedef struct otRadioIeInfo otRadioIeInfo
```

Represents the IEEE 802.15.4 Header IE (Information Element) related information of a radio frame.

Definition at line 249 of file include/openthread/platform/radio.h

otRadioFrame

```
typedef struct otRadioFrame otRadioFrame
```

Represents an IEEE 802.15.4 radio frame.

Definition at line 383 of file include/openthread/platform/radio.h

otRadioState

```
typedef enum otRadioState otRadioState
```

Represents the state of a radio.

Initially, a radio is in the Disabled state.

Definition at line 396 of file include/openthread/platform/radio.h

otRadioCoexMetrics

```
typedef struct otRadioCoexMetrics otRadioCoexMetrics
```

The following are valid radio state transitions:

```
                                (Radio ON)

+---------+ Enable() +------+ Receive() +--------+ Transmit() +---------+ | |---------->| |---------->| |---------->| | |
Disabled | | Sleep | | Receive | | Transmit | | |<----------| |<----------| |<----------| | +---------+ Disable() +------+
Sleep() +--------+ Receive() +---------+ (Radio OFF) or signal TransmitDone
```

During the IEEE 802.15.4 data request command the transition Sleep->Receive->Transmit can be shortened to direct transition from Sleep to Transmit if the platform supports the OT_RADIO_CAPS_SLEEP_TO_TX capability. Represents radio coexistence metrics.

Definition at line 439 of file include/openthread/platform/radio.h

otLinkMetrics

```
typedef struct otLinkMetrics otLinkMetrics
```

Represents what metrics are specified to query.

Definition at line 452 of file include/openthread/platform/radio.h

Variable Documentation

OT_TOOL_PACKED_END

```
OT_TOOL_PACKED_BEGIN struct otMacKey OT_TOOL_PACKED_END
```

Definition at line 182 of file include/openthread/platform/radio.h

Macro Definition Documentation

OT_PANID_BROADCAST

```
#define OT_PANID_BROADCAST
```

Value:

```
0xffff
```

IEEE 802.15.4 Broadcast PAN ID.

Definition at line 141 of file include/openthread/platform/radio.h

OT_EXT_ADDRESS_SIZE

```
#define OT_EXT_ADDRESS_SIZE
```

Value:

```
8
```

Size of an IEEE 802.15.4 Extended Address (bytes)

Definition at line 155 of file include/openthread/platform/radio.h

CSL_IE_HEADER_BYTES_LO

```
#define CSL_IE_HEADER_BYTES_LO
```

Value:

```
0x04
```

Fixed CSL IE header first byte.

Definition at line 169 of file include/openthread/platform/radio.h

CSL_IE_HEADER_BYTES_HI

```
#define CSL_IE_HEADER_BYTES_HI
```

Value:

```
0x0d
```

Fixed CSL IE header second byte.

Definition at line 170 of file include/openthread/platform/radio.h

OT_MAC_KEY_SIZE

```
#define OT_MAC_KEY_SIZE
```

Value:

```
16
```

Size of the MAC Key in bytes.

Definition at line 190 of file include/openthread/platform/radio.h

OT_TOOL_PACKED_END

```
#define OT_TOOL_PACKED_END
```

Compiler-specific indication at the end of a byte packed class or struct.

Definition at line 178 of file include/openthread/platform/toolchain.h

otExtAddress

Represents the IEEE 802.15.4 Extended Address.

Public Attributes

uint8_t	m8 IEEE 802.15.4 Extended Address bytes.

Public Attribute Documentation

m8

```
uint8_t otExtAddress::m8[OT_EXT_ADDRESS_SIZE]
```

IEEE 802.15.4 Extended Address bytes.

Definition at line 181 of file include/openthread/platform/radio.h

otMacKey

Represents a MAC Key.

Public Attributes

uint8_t	m8 MAC Key bytes.

Public Attribute Documentation

m8

```
uint8_t otMacKey::m8[OT_MAC_KEY_SIZE]
```

MAC Key bytes.

Definition at line 201 of file include/openthread/platform/radio.h

otMacKeyMaterial

Represents a MAC Key.

Public Attributes

otMacKeyRef	mKeyRef Reference to the key stored.
otMacKey	mKey Key stored as literal.
union otMacKeyMaterial ::@17	mKeyMaterial

Public Attribute Documentation

mKeyRef

```
otMacKeyRef otMacKeyMaterial::mKeyRef
```

Reference to the key stored.

Definition at line 226 of file include/openthread/platform/radio.h

mKey

```
otMacKey otMacKeyMaterial::mKey
```

Key stored as literal.

Definition at line 227 of file include/openthread/platform/radio.h

mKeyMaterial

```
union otMacKeyMaterial::@17 otMacKeyMaterial::mKeyMaterial
```

Definition at line 228 of file include/openthread/platform/radio.h

otRadioIeInfo

Represents the IEEE 802.15.4 Header IE (Information Element) related information of a radio frame.

Public Attributes

int64_t	mNetworkTimeOffset The time offset to the Thread network time.
uint8_t	mTimeIeOffset The Time IE offset from the start of PSDU.
uint8_t	mTimeSyncSeq The Time sync sequence.

Public Attribute Documentation

mNetworkTimeOffset

```
int64_t otRadioIeInfo::mNetworkTimeOffset
```

The time offset to the Thread network time.

Definition at line 246 of file include/openthread/platform/radio.h

mTimeIeOffset

```
uint8_t otRadioIeInfo::mTimeIeOffset
```

The Time IE offset from the start of PSDU.

Definition at line 247 of file include/openthread/platform/radio.h

mTimeSyncSeq

```
uint8_t otRadioIeInfo::mTimeSyncSeq
```

The Time sync sequence.

Definition at line 248 of file include/openthread/platform/radio.h

otRadioFrame

Represents an IEEE 802.15.4 radio frame.

Public Attributes

uint8_t *	mPsdu The PSDU.
uint16_t	mLength Length of the PSDU.
uint8_t	mChannel Channel used to transmit/receive the frame.
uint8_t	mRadioType Radio link type - should be ignored by radio driver.
uint8_t	mIid Interface Id for the incoming/outgoing radio packet.
const otMacKeyMaterial *	mAesKey The key material used for AES-CCM frame security.
otRadioIeInfo *	mIeInfo The pointer to the Header IE(s) related information.
uint32_t	mTxDelayBaseTime The base time in microseconds for scheduled transmissions relative to the local radio clock, see `otPlatRadioGetNow` and `mTxDelay`.
uint32_t	mTxDelay The delay time in microseconds for this transmission referenced to `mTxDelayBaseTime`.
uint8_t	mMaxCsmaBackoffs Maximum number of backoffs attempts before declaring CCA failure.
uint8_t	mMaxFrameRetries Maximum number of retries allowed after a transmission failure.
uint8_t	mRxChannelAfterTxDone The RX channel after frame TX is done (after all frame retries - ack received, or timeout, or abort).
bool	mIsHeaderUpdated Indicates whether frame counter and CSL IEs are properly updated in the header.
bool	mIsARetx Indicates whether the frame is a retransmission or not.
bool	mCsmaCaEnabled Set to true to enable CSMA-CA for this packet, false otherwise.
bool	mCslPresent Set to true if CSL header IE is present.

bool	mIsSecurityProcessed True if SubMac should skip the AES processing of this frame.
struct otRadioFrame::@18::@19	mTxInfo Structure representing radio frame transmit information.
uint64_t	mTimestamp The time of the local radio clock in microseconds when the end of the SFD was present at the local antenna.
uint32_t	mAckFrameCounter ACK security frame counter (applicable when `mAckedWithSecEnhAck` is set).
uint8_t	mAckKeyId ACK security key index (applicable when `mAckedWithSecEnhAck` is set).
int8_t	mRssi Received signal strength indicator in dBm for received frames.
uint8_t	mLqi Link Quality Indicator for received frames.
bool	mAckedWithFramePending This indicates if this frame was acknowledged with frame pending set.
bool	mAckedWithSecEnhAck This indicates if this frame was acknowledged with secured enhance ACK.
struct otRadioFrame::@18::@20	mRxInfo Structure representing radio frame receive information.
union otRadioFrame::@18	mInfo The union of transmit and receive information for a radio frame.

Public Attribute Documentation

mPsdu

```
uint8_t* otRadioFrame::mPsdu
```

The PSDU.

Definition at line `256` of file `include/openthread/platform/radio.h`

mLength

```
uint16_t otRadioFrame::mLength
```

Length of the PSDU.

Definition at line `258` of file `include/openthread/platform/radio.h`

mChannel

uint8_t otRadioFrame::mChannel

Channel used to transmit/receive the frame.

Definition at line 259 of file include/openthread/platform/radio.h

mRadioType

uint8_t otRadioFrame::mRadioType

Radio link type - should be ignored by radio driver.

Definition at line 261 of file include/openthread/platform/radio.h

mIid

uint8_t otRadioFrame::mIid

Interface Id for the incoming/outgoing radio packet.

This field is used by RCP when OPENTHREAD_CONFIG_MULTIPAN_RCP_ENABLE is enabled. If not enabled, this field defaults to zero.

Incoming packets are marked with the correct IID to deliver to the appropriate host. RCP determines the IID value based on the destination PAN ID of the received packet. The IID value of zero indicates the broadcast packet, which will send it to all the hosts.

For outgoing packets, IID is used to format the transmit done callback for the appropriate host. RCP extracts the IID value while processing the received spinel frame from the spinel header.

Definition at line 278 of file include/openthread/platform/radio.h

mAesKey

const otMacKeyMaterial* otRadioFrame::mAesKey

The key material used for AES-CCM frame security.

Definition at line 290 of file include/openthread/platform/radio.h

mIeInfo

otRadioIeInfo* otRadioFrame::mIeInfo

The pointer to the Header IE(s) related information.

Definition at line 291 of file include/openthread/platform/radio.h

mTxDelayBaseTime

uint32_t otRadioFrame::mTxDelayBaseTime

The base time in microseconds for scheduled transmissions relative to the local radio clock, see `otPlatRadioGetNow` and `mTxDelay` .

Definition at line `298` of file `include/openthread/platform/radio.h`

mTxDelay

uint32_t otRadioFrame::mTxDelay

The delay time in microseconds for this transmission referenced to `mTxDelayBaseTime` .

Note: `mTxDelayBaseTime` + `mTxDelay` SHALL point to the point in time when the end of the SFD will be present at the local antenna, relative to the local radio clock.

Definition at line `308` of file `include/openthread/platform/radio.h`

mMaxCsmaBackoffs

uint8_t otRadioFrame::mMaxCsmaBackoffs

Maximum number of backoffs attempts before declaring CCA failure.

Definition at line `310` of file `include/openthread/platform/radio.h`

mMaxFrameRetries

uint8_t otRadioFrame::mMaxFrameRetries

Maximum number of retries allowed after a transmission failure.

Definition at line `311` of file `include/openthread/platform/radio.h`

mRxChannelAfterTxDone

uint8_t otRadioFrame::mRxChannelAfterTxDone

The RX channel after frame TX is done (after all frame retries - ack received, or timeout, or abort).

Radio platforms can choose to fully ignore this. OT stack will make sure to call `otPlatRadioReceive()` with the desired RX channel after a frame TX is done and signaled in `otPlatRadioTxDone()` callback. Radio platforms that don't provide `OT_RADIO_CAPS_TRANSMIT_RETRIES` must always ignore this.

This is intended for situations where there may be delay in interactions between OT stack and radio, as an example this is used in RCP/host architecture to make sure RCP switches to PAN channel more quickly. In particular, this can help with CSL tx to a sleepy child, where the child may use a different channel for CSL than the PAN channel. After frame tx, we want the radio/RCP to go back to the PAN channel quickly to ensure that parent does not miss tx from child afterwards, e.g., child responding to the earlier CSL transmitted frame from parent using PAN channel while radio still staying on CSL channel.

The switch to the RX channel MUST happen after the frame TX is fully done, i.e., after all retries and when ack is received (when "Ack Request" flag is set on the TX frame) or ack timeout. Note that ack is expected on the same channel that

frame is sent on.

Definition at line 332 of file include/openthread/platform/radio.h

mIsHeaderUpdated

```
bool otRadioFrame::mIsHeaderUpdated
```

Indicates whether frame counter and CSL IEs are properly updated in the header.

If the platform layer does not provide OT_RADIO_CAPS_TRANSMIT_SEC capability, it can ignore this flag.

If the platform provides OT_RADIO_CAPS_TRANSMIT_SEC capability, then platform is expected to handle tx security processing and assignment of frame counter. In this case the following behavior is expected:

When mIsHeaderUpdated is set, it indicates that OpenThread core has already set the frame counter and CSL IEs (if security is enabled) in the prepared frame. The counter is ensured to match the counter value from the previous attempts of the same frame. The platform should not assign or change the frame counter (but may still need to perform security processing depending on mIsSecurityProcessed flag).

If mIsHeaderUpdated is not set, then the frame counter and key CSL IE not set in the frame by OpenThread core and it is the responsibility of the radio platform to assign them. The platform must update the frame header (assign counter and CSL IE values) before sending the frame over the air, however if the the transmission gets aborted and the frame is never sent over the air (e.g., channel access error) the platform may choose to not update the header. If the platform updates the header, it must also set this flag before passing the frame back from the otPlatRadioTxDone() callback.

Definition at line 355 of file include/openthread/platform/radio.h

mIsARetx

```
bool otRadioFrame::mIsARetx
```

Indicates whether the frame is a retransmission or not.

Definition at line 356 of file include/openthread/platform/radio.h

mCsmaCaEnabled

```
bool otRadioFrame::mCsmaCaEnabled
```

Set to true to enable CSMA-CA for this packet, false otherwise.

Definition at line 357 of file include/openthread/platform/radio.h

mCslPresent

```
bool otRadioFrame::mCslPresent
```

Set to true if CSL header IE is present.

Definition at line 358 of file include/openthread/platform/radio.h

mIsSecurityProcessed

```
bool otRadioFrame::mIsSecurityProcessed
```

True if SubMac should skip the AES processing of this frame.

Definition at line `359` of file `include/openthread/platform/radio.h`

mTxInfo

```
struct otRadioFrame::@18::@19 otRadioFrame::mTxInfo
```

Structure representing radio frame transmit information.

Definition at line `360` of file `include/openthread/platform/radio.h`

mTimestamp

```
uint64_t otRadioFrame::mTimestamp
```

The time of the local radio clock in microseconds when the end of the SFD was present at the local antenna.

Definition at line `371` of file `include/openthread/platform/radio.h`

mAckFrameCounter

```
uint32_t otRadioFrame::mAckFrameCounter
```

ACK security frame counter (applicable when `mAckedWithSecEnhAck` is set).

Definition at line `373` of file `include/openthread/platform/radio.h`

mAckKeyId

```
uint8_t otRadioFrame::mAckKeyId
```

ACK security key index (applicable when `mAckedWithSecEnhAck` is set).

Definition at line `374` of file `include/openthread/platform/radio.h`

mRssi

```
int8_t otRadioFrame::mRssi
```

Received signal strength indicator in dBm for received frames.

Definition at line `375` of file `include/openthread/platform/radio.h`

mLqi

uint8_t otRadioFrame::mLqi

Link Quality Indicator for received frames.

Definition at line 376 of file include/openthread/platform/radio.h

mAckedWithFramePending

bool otRadioFrame::mAckedWithFramePending

This indicates if this frame was acknowledged with frame pending set.

Definition at line 379 of file include/openthread/platform/radio.h

mAckedWithSecEnhAck

bool otRadioFrame::mAckedWithSecEnhAck

This indicates if this frame was acknowledged with secured enhance ACK.

Definition at line 380 of file include/openthread/platform/radio.h

mRxInfo

struct otRadioFrame::@18::@20 otRadioFrame::mRxInfo

Structure representing radio frame receive information.

Definition at line 381 of file include/openthread/platform/radio.h

mInfo

union otRadioFrame::@18 otRadioFrame::mInfo

The union of transmit and receive information for a radio frame.

Definition at line 382 of file include/openthread/platform/radio.h

otRadioCoexMetrics

The following are valid radio state transitions:

(Radio ON)

+---------—+ Enable() +------—+ Receive() +--------—+ Transmit() +---------—+ | |----------—>| |----------—>| |------------—>| | | Disabled | | Sleep | | Receive | | Transmit | | |<----------—| |<----------—| |<------------—| | +---------—+ Disable() +------—+ Sleep() +--------—+ Receive() +---------—+ (Radio OFF) or signal TransmitDone

During the IEEE 802.15.4 data request command the transition Sleep->Receive->Transmit can be shortened to direct transition from Sleep to Transmit if the platform supports the OT_RADIO_CAPS_SLEEP_TO_TX capability. Represents radio coexistence metrics.

Public Attributes

uint32_t **mNumGrantGlitch**
Number of grant glitches.

uint32_t **mNumTxRequest**
Number of tx requests.

uint32_t **mNumTxGrantImmediate**
Number of tx requests while grant was active.

uint32_t **mNumTxGrantWait**
Number of tx requests while grant was inactive.

uint32_t **mNumTxGrantWaitActivated**
Number of tx requests while grant was inactive that were ultimately granted.

uint32_t **mNumTxGrantWaitTimeout**
Number of tx requests while grant was inactive that timed out.

uint32_t **mNumTxGrantDeactivatedDuringRequest**
Number of tx that were in progress when grant was deactivated.

uint32_t **mNumTxDelayedGrant**
Number of tx requests that were not granted within 50us.

uint32_t **mAvgTxRequestToGrantTime**
Average time in usec from tx request to grant.

uint32_t **mNumRxRequest**
Number of rx requests.

uint32_t **mNumRxGrantImmediate**
Number of rx requests while grant was active.

uint32_t **mNumRxGrantWait**
Number of rx requests while grant was inactive.

uint32_t **mNumRxGrantWaitActivated**
Number of rx requests while grant was inactive that were ultimately granted.

uint32_t	mNumRxGrantWaitTimeout Number of rx requests while grant was inactive that timed out.
uint32_t	mNumRxGrantDeactivatedDuringRequest Number of rx that were in progress when grant was deactivated.
uint32_t	mNumRxDelayedGrant Number of rx requests that were not granted within 50us.
uint32_t	mAvgRxRequestToGrantTime Average time in usec from rx request to grant.
uint32_t	mNumRxGrantNone Number of rx requests that completed without receiving grant.
bool	mStopped Stats collection stopped due to saturation.

Public Attribute Documentation

mNumGrantGlitch

```
uint32_t otRadioCoexMetrics::mNumGrantGlitch
```

Number of grant glitches.

Definition at line 420 of file include/openthread/platform/radio.h

mNumTxRequest

```
uint32_t otRadioCoexMetrics::mNumTxRequest
```

Number of tx requests.

Definition at line 421 of file include/openthread/platform/radio.h

mNumTxGrantImmediate

```
uint32_t otRadioCoexMetrics::mNumTxGrantImmediate
```

Number of tx requests while grant was active.

Definition at line 422 of file include/openthread/platform/radio.h

mNumTxGrantWait

```
uint32_t otRadioCoexMetrics::mNumTxGrantWait
```

Number of tx requests while grant was inactive.

Definition at line 423 of file include/openthread/platform/radio.h

mNumTxGrantWaitActivated

uint32_t otRadioCoexMetrics::mNumTxGrantWaitActivated

Number of tx requests while grant was inactive that were ultimately granted.

Definition at line 424 of file include/openthread/platform/radio.h

mNumTxGrantWaitTimeout

uint32_t otRadioCoexMetrics::mNumTxGrantWaitTimeout

Number of tx requests while grant was inactive that timed out.

Definition at line 425 of file include/openthread/platform/radio.h

mNumTxGrantDeactivatedDuringRequest

uint32_t otRadioCoexMetrics::mNumTxGrantDeactivatedDuringRequest

Number of tx that were in progress when grant was deactivated.

Definition at line 426 of file include/openthread/platform/radio.h

mNumTxDelayedGrant

uint32_t otRadioCoexMetrics::mNumTxDelayedGrant

Number of tx requests that were not granted within 50us.

Definition at line 427 of file include/openthread/platform/radio.h

mAvgTxRequestToGrantTime

uint32_t otRadioCoexMetrics::mAvgTxRequestToGrantTime

Average time in usec from tx request to grant.

Definition at line 428 of file include/openthread/platform/radio.h

mNumRxRequest

uint32_t otRadioCoexMetrics::mNumRxRequest

Number of rx requests.

Definition at line 429 of file include/openthread/platform/radio.h

mNumRxGrantImmediate

uint32_t otRadioCoexMetrics::mNumRxGrantImmediate

Number of rx requests while grant was active.

Definition at line 430 of file include/openthread/platform/radio.h

mNumRxGrantWait

uint32_t otRadioCoexMetrics::mNumRxGrantWait

Number of rx requests while grant was inactive.

Definition at line 431 of file include/openthread/platform/radio.h

mNumRxGrantWaitActivated

uint32_t otRadioCoexMetrics::mNumRxGrantWaitActivated

Number of rx requests while grant was inactive that were ultimately granted.

Definition at line 432 of file include/openthread/platform/radio.h

mNumRxGrantWaitTimeout

uint32_t otRadioCoexMetrics::mNumRxGrantWaitTimeout

Number of rx requests while grant was inactive that timed out.

Definition at line 433 of file include/openthread/platform/radio.h

mNumRxGrantDeactivatedDuringRequest

uint32_t otRadioCoexMetrics::mNumRxGrantDeactivatedDuringRequest

Number of rx that were in progress when grant was deactivated.

Definition at line 434 of file include/openthread/platform/radio.h

mNumRxDelayedGrant

uint32_t otRadioCoexMetrics::mNumRxDelayedGrant

Number of rx requests that were not granted within 50us.

Definition at line 435 of file include/openthread/platform/radio.h

mAvgRxRequestToGrantTime

> uint32_t otRadioCoexMetrics::mAvgRxRequestToGrantTime

Average time in usec from rx request to grant.

Definition at line `436` of file `include/openthread/platform/radio.h`

mNumRxGrantNone

> uint32_t otRadioCoexMetrics::mNumRxGrantNone

Number of rx requests that completed without receiving grant.

Definition at line `437` of file `include/openthread/platform/radio.h`

mStopped

> bool otRadioCoexMetrics::mStopped

Stats collection stopped due to saturation.

Definition at line `438` of file `include/openthread/platform/radio.h`

otLinkMetrics

Represents what metrics are specified to query.

Public Attributes

bool	mPduCount Pdu count.
bool	mLqi Link Quality Indicator.
bool	mLinkMargin Link Margin.
bool	mRssi Received Signal Strength Indicator.
bool	mReserved Reserved, this is for reference device.

Public Attribute Documentation

mPduCount

```
bool otLinkMetrics::mPduCount
```

Pdu count.

Definition at line 447 of file include/openthread/platform/radio.h

mLqi

```
bool otLinkMetrics::mLqi
```

Link Quality Indicator.

Definition at line 448 of file include/openthread/platform/radio.h

mLinkMargin

```
bool otLinkMetrics::mLinkMargin
```

Link Margin.

Definition at line 449 of file include/openthread/platform/radio.h

mRssi

```
bool otLinkMetrics::mRssi
```

Received Signal Strength Indicator.

Definition at line 450 of file include/openthread/platform/radio.h

mReserved

```
bool otLinkMetrics::mReserved
```

Reserved, this is for reference device.

Definition at line 451 of file include/openthread/platform/radio.h

Radio Configuration

Radio Configuration

This module includes the platform abstraction for radio configuration.

Functions

otRadioCaps	otPlatRadioGetCaps(otInstance *aInstance) Get the radio capabilities.
const char *	otPlatRadioGetVersionString(otInstance *aInstance) Get the radio version string.
int8_t	otPlatRadioGetReceiveSensitivity(otInstance *aInstance) Get the radio receive sensitivity value.
void	otPlatRadioGetIeeeEui64(otInstance *aInstance, uint8_t *aIeeeEui64) Gets the factory-assigned IEEE EUI-64 for this interface.
void	otPlatRadioSetPanId(otInstance *aInstance, otPanId aPanId) Set the PAN ID for address filtering.
void	otPlatRadioSetExtendedAddress(otInstance *aInstance, const otExtAddress *aExtAddress) Set the Extended Address for address filtering.
void	otPlatRadioSetShortAddress(otInstance *aInstance, otShortAddress aShortAddress) Set the Short Address for address filtering.
otError	otPlatRadioGetTransmitPower(otInstance *aInstance, int8_t *aPower) Get the radio's transmit power in dBm.
otError	otPlatRadioSetTransmitPower(otInstance *aInstance, int8_t aPower) Set the radio's transmit power in dBm.
otError	otPlatRadioGetCcaEnergyDetectThreshold(otInstance *aInstance, int8_t *aThreshold) Get the radio's CCA ED threshold in dBm measured at antenna connector per IEEE 802.15.4 - 2015 section 10.1.4.
otError	otPlatRadioSetCcaEnergyDetectThreshold(otInstance *aInstance, int8_t aThreshold) Set the radio's CCA ED threshold in dBm measured at antenna connector per IEEE 802.15.4 - 2015 section 10.1.4.
otError	otPlatRadioGetFemLnaGain(otInstance *aInstance, int8_t *aGain) Gets the external FEM's Rx LNA gain in dBm.
otError	otPlatRadioSetFemLnaGain(otInstance *aInstance, int8_t aGain) Sets the external FEM's Rx LNA gain in dBm.
bool	otPlatRadioGetPromiscuous(otInstance *aInstance) Get the status of promiscuous mode.
void	otPlatRadioSetPromiscuous(otInstance *aInstance, bool aEnable) Enable or disable promiscuous mode.

void otPlatRadioSetMacKey(otInstance *aInstance, uint8_t aKeyIdMode, uint8_t aKeyId, const otMacKeyMaterial *aPrevKey, const otMacKeyMaterial *aCurrKey, const otMacKeyMaterial *aNextKey, otRadioKeyType aKeyType)
Update MAC keys and key index.

void otPlatRadioSetMacFrameCounter(otInstance *aInstance, uint32_t aMacFrameCounter)
Sets the current MAC frame counter value.

void otPlatRadioSetMacFrameCounterIfLarger(otInstance *aInstance, uint32_t aMacFrameCounter)
Sets the current MAC frame counter value only if the new given value is larger than the current value.

uint64_t otPlatRadioGetNow(otInstance *aInstance)
Get the current time in microseconds referenced to a continuous monotonic local radio clock (64 bits width).

uint32_t otPlatRadioGetBusSpeed(otInstance *aInstance)
Get the bus speed in bits/second between the host and the radio chip.

Function Documentation

otPlatRadioGetCaps

```
otRadioCaps otPlatRadioGetCaps (otInstance *aInstance)
```

Get the radio capabilities.

Parameters

[in]	aInstance	The OpenThread instance structure.

Returns

- The radio capability bit vector (see `OT_RADIO_CAP_*` definitions).

Definition at line `477` of file `include/openthread/platform/radio.h`

otPlatRadioGetVersionString

```
const char * otPlatRadioGetVersionString (otInstance *aInstance)
```

Get the radio version string.

Parameters

[in]	aInstance	The OpenThread instance structure.

This is an optional radio driver platform function. If not provided by platform radio driver, OpenThread uses the OpenThread version instead (**See Also**

- otGetVersionString()).

Returns

- A pointer to the OpenThread radio version.

Definition at line `490` of file `include/openthread/platform/radio.h`

otPlatRadioGetReceiveSensitivity

```
int8_t otPlatRadioGetReceiveSensitivity (otInstance *aInstance)
```

Get the radio receive sensitivity value.

Parameters

[in]	aInstance	The OpenThread instance structure.

Returns

- The radio receive sensitivity value in dBm.

Definition at line `500` of file `include/openthread/platform/radio.h`

otPlatRadioGetIeeeEui64

```
void otPlatRadioGetIeeeEui64 (otInstance *aInstance, uint8_t *aIeeeEui64)
```

Gets the factory-assigned IEEE EUI-64 for this interface.

Parameters

[in]	aInstance	The OpenThread instance structure.
[out]	aIeeeEui64	A pointer to the factory-assigned IEEE EUI-64.

Definition at line `509` of file `include/openthread/platform/radio.h`

otPlatRadioSetPanId

```
void otPlatRadioSetPanId (otInstance *aInstance, otPanId aPanId)
```

Set the PAN ID for address filtering.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aPanId	The IEEE 802.15.4 PAN ID.

Definition at line `518` of file `include/openthread/platform/radio.h`

otPlatRadioSetExtendedAddress

```
void otPlatRadioSetExtendedAddress (otInstance *aInstance, const otExtAddress *aExtAddress)
```

Set the Extended Address for address filtering.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aExtAddress	A pointer to the IEEE 802.15.4 Extended Address stored in little-endian byte order.

Definition at line `528` of file `include/openthread/platform/radio.h`

otPlatRadioSetShortAddress

```
void otPlatRadioSetShortAddress (otInstance *aInstance, otShortAddress aShortAddress)
```

Set the Short Address for address filtering.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aShortAddress	The IEEE 802.15.4 Short Address.

Definition at line `537` of file `include/openthread/platform/radio.h`

otPlatRadioGetTransmitPower

> otError otPlatRadioGetTransmitPower (otInstance *aInstance, int8_t *aPower)

Get the radio's transmit power in dBm.

Parameters

[in]	aInstance	The OpenThread instance structure.
[out]	aPower	The transmit power in dBm.

Note

- The transmit power returned will be no larger than the power specified in the max power table for the current channel.

Definition at line `553` of file `include/openthread/platform/radio.h`

otPlatRadioSetTransmitPower

> otError otPlatRadioSetTransmitPower (otInstance *aInstance, int8_t aPower)

Set the radio's transmit power in dBm.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aPower	The transmit power in dBm.

Note

- The real transmit power will be no larger than the power specified in the max power table for the current channel.

Definition at line `568` of file `include/openthread/platform/radio.h`

otPlatRadioGetCcaEnergyDetectThreshold

> otError otPlatRadioGetCcaEnergyDetectThreshold (otInstance *aInstance, int8_t *aThreshold)

Get the radio's CCA ED threshold in dBm measured at antenna connector per IEEE 802.15.4 - 2015 section 10.1.4.

Parameters

[in]	aInstance	The OpenThread instance structure.
[out]	aThreshold	The CCA ED threshold in dBm.

Definition at line `581` of file `include/openthread/platform/radio.h`

otPlatRadioSetCcaEnergyDetectThreshold

> otError otPlatRadioSetCcaEnergyDetectThreshold (otInstance *aInstance, int8_t aThreshold)

Set the radio's CCA ED threshold in dBm measured at antenna connector per IEEE 802.15.4 - 2015 section 10.1.4.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aThreshold	The CCA ED threshold in dBm.

Definition at line `594` of file `include/openthread/platform/radio.h`

otPlatRadioGetFemLnaGain

> otError otPlatRadioGetFemLnaGain (otInstance *aInstance, int8_t *aGain)

Gets the external FEM's Rx LNA gain in dBm.

Parameters

[in]	aInstance	The OpenThread instance structure.
[out]	aGain	The external FEM's Rx LNA gain in dBm.

Definition at line `607` of file `include/openthread/platform/radio.h`

otPlatRadioSetFemLnaGain

> otError otPlatRadioSetFemLnaGain (otInstance *aInstance, int8_t aGain)

Sets the external FEM's Rx LNA gain in dBm.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aGain	The external FEM's Rx LNA gain in dBm.

Definition at line `619` of file `include/openthread/platform/radio.h`

otPlatRadioGetPromiscuous

> bool otPlatRadioGetPromiscuous (otInstance *aInstance)

Get the status of promiscuous mode.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `630` of file `include/openthread/platform/radio.h`

otPlatRadioSetPromiscuous

> void otPlatRadioSetPromiscuous (otInstance *aInstance, bool aEnable)

Enable or disable promiscuous mode.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aEnable	TRUE to enable or FALSE to disable promiscuous mode.

Definition at line `639` of file `include/openthread/platform/radio.h`

otPlatRadioSetMacKey

> void otPlatRadioSetMacKey (otInstance *aInstance, uint8_t aKeyIdMode, uint8_t aKeyId, const otMacKeyMaterial *aPrevKey, const otMacKeyMaterial *aCurrKey, const otMacKeyMaterial *aNextKey, otRadioKeyType aKeyType)

Update MAC keys and key index.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aKeyIdMode	The key ID mode.
[in]	aKeyId	Current MAC key index.
[in]	aPrevKey	A pointer to the previous MAC key.
[in]	aCurrKey	A pointer to the current MAC key.
[in]	aNextKey	A pointer to the next MAC key.
[in]	aKeyType	Key Type used.

Is used when radio provides OT_RADIO_CAPS_TRANSMIT_SEC capability.

Definition at line `655` of file `include/openthread/platform/radio.h`

otPlatRadioSetMacFrameCounter

> void otPlatRadioSetMacFrameCounter (otInstance *aInstance, uint32_t aMacFrameCounter)

Sets the current MAC frame counter value.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMacFrameCounter	The MAC frame counter value.

Is used when radio provides `OT_RADIO_CAPS_TRANSMIT_SEC` capability.

Definition at line `672` of file `include/openthread/platform/radio.h`

otPlatRadioSetMacFrameCounterIfLarger

> void otPlatRadioSetMacFrameCounterIfLarger (otInstance *aInstance, uint32_t aMacFrameCounter)

Sets the current MAC frame counter value only if the new given value is larger than the current value.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.
[in]	aMacFrameCounter	The MAC frame counter value.

Is used when radio provides `OT_RADIO_CAPS_TRANSMIT_SEC` capability.

Definition at line `683` of file `include/openthread/platform/radio.h`

otPlatRadioGetNow

```
uint64_t otPlatRadioGetNow (otInstance *aInstance)
```

Get the current time in microseconds referenced to a continuous monotonic local radio clock (64 bits width).

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

The radio clock SHALL NOT wrap during the device's uptime. Implementations SHALL therefore identify and compensate for internal counter overflows. The clock does not have a defined epoch and it SHALL NOT introduce any continuous or discontinuous adjustments (e.g. leap seconds). Implementations SHALL compensate for any sleep times of the device.

Implementations MAY choose to discipline the radio clock and compensate for sleep times by any means (e.g. by combining a high precision/low power RTC with a high resolution counter) as long as the exposed combined clock provides continuous monotonic microsecond resolution ticks within the accuracy limits announced by otPlatRadioGetCslAccuracy.

Returns

- The current time in microseconds. UINT64_MAX when platform does not support or radio time is not ready.

Definition at line `707` of file `include/openthread/platform/radio.h`

otPlatRadioGetBusSpeed

```
uint32_t otPlatRadioGetBusSpeed (otInstance *aInstance)
```

Get the bus speed in bits/second between the host and the radio chip.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

Returns

- The bus speed in bits/second between the host and the radio chip. Return 0 when the MAC and above layer and Radio layer resides on the same chip.

Definition at line `718` of file `include/openthread/platform/radio.h`

Radio Operation

Radio Operation

This module includes the platform abstraction for radio operations.

Functions

otRadioState	otPlatRadioGetState(otInstance *aInstance) Get current state of the radio.
otError	otPlatRadioEnable(otInstance *aInstance) Enable the radio.
otError	otPlatRadioDisable(otInstance *aInstance) Disable the radio.
bool	otPlatRadioIsEnabled(otInstance *aInstance) Check whether radio is enabled or not.
otError	otPlatRadioSleep(otInstance *aInstance) Transition the radio from Receive to Sleep (turn off the radio).
otError	otPlatRadioReceive(otInstance *aInstance, uint8_t aChannel) Transition the radio from Sleep to Receive (turn on the radio).
otError	otPlatRadioReceiveAt(otInstance *aInstance, uint8_t aChannel, uint32_t aStart, uint32_t aDuration) Schedule a radio reception window at a specific time and duration.
void	otPlatRadioReceiveDone(otInstance *aInstance, otRadioFrame *aFrame, otError aError) The radio driver calls this method to notify OpenThread of a received frame.
void	otPlatDiagRadioReceiveDone(otInstance *aInstance, otRadioFrame *aFrame, otError aError) The radio driver calls this method to notify OpenThread diagnostics module of a received frame.
otRadioFrame *	otPlatRadioGetTransmitBuffer(otInstance *aInstance) Get the radio transmit frame buffer.
otError	otPlatRadioTransmit(otInstance *aInstance, otRadioFrame *aFrame) Begin the transmit sequence on the radio.
void	otPlatRadioTxStarted(otInstance *aInstance, otRadioFrame *aFrame) The radio driver calls this method to notify OpenThread that the transmission has started.
void	otPlatRadioTxDone(otInstance *aInstance, otRadioFrame *aFrame, otRadioFrame *aAckFrame, otError aError) The radio driver calls this function to notify OpenThread that the transmit operation has completed, providing both the transmitted frame and, if applicable, the received ack frame.
void	otPlatDiagRadioTransmitDone(otInstance *aInstance, otRadioFrame *aFrame, otError aError) The radio driver calls this method to notify OpenThread diagnostics module that the transmission has completed.
int8_t	otPlatRadioGetRssi(otInstance *aInstance) Get the most recent RSSI measurement.

otError otPlatRadioEnergyScan(otInstance *aInstance, uint8_t aScanChannel, uint16_t aScanDuration)
Begin the energy scan sequence on the radio.

void otPlatRadioEnergyScanDone(otInstance *aInstance, int8_t aEnergyScanMaxRssi)
The radio driver calls this method to notify OpenThread that the energy scan is complete.

void otPlatRadioEnableSrcMatch(otInstance *aInstance, bool aEnable)
Enable/Disable source address match feature.

otError otPlatRadioAddSrcMatchShortEntry(otInstance *aInstance, otShortAddress aShortAddress)
Add a short address to the source address match table.

otError otPlatRadioAddSrcMatchExtEntry(otInstance *aInstance, const otExtAddress *aExtAddress)
Add an extended address to the source address match table.

otError otPlatRadioClearSrcMatchShortEntry(otInstance *aInstance, otShortAddress aShortAddress)
Remove a short address from the source address match table.

otError otPlatRadioClearSrcMatchExtEntry(otInstance *aInstance, const otExtAddress *aExtAddress)
Remove an extended address from the source address match table.

void otPlatRadioClearSrcMatchShortEntries(otInstance *aInstance)
Clear all short addresses from the source address match table.

void otPlatRadioClearSrcMatchExtEntries(otInstance *aInstance)
Clear all the extended/long addresses from source address match table.

uint32_t otPlatRadioGetSupportedChannelMask(otInstance *aInstance)
Get the radio supported channel mask that the device is allowed to be on.

uint32_t otPlatRadioGetPreferredChannelMask(otInstance *aInstance)
Gets the radio preferred channel mask that the device prefers to form on.

otError otPlatRadioSetCoexEnabled(otInstance *aInstance, bool aEnabled)
Enable the radio coex.

bool otPlatRadioIsCoexEnabled(otInstance *aInstance)
Check whether radio coex is enabled or not.

otError otPlatRadioGetCoexMetrics(otInstance *aInstance, otRadioCoexMetrics *aCoexMetrics)
Get the radio coexistence metrics.

otError otPlatRadioEnableCsl(otInstance *aInstance, uint32_t aCslPeriod, otShortAddress aShortAddr, const otExtAddress *aExtAddr)
Enable or disable CSL receiver.

void otPlatRadioUpdateCslSampleTime(otInstance *aInstance, uint32_t aCslSampleTime)
Update CSL sample time in radio driver.

uint8_t otPlatRadioGetCslAccuracy(otInstance *aInstance)
Get the current estimated worst case accuracy (maximum ± deviation from the nominal frequency) of the local radio clock in units of PPM.

uint8_t otPlatRadioGetCslUncertainty(otInstance *aInstance)
The fixed uncertainty (i.e.

otError otPlatRadioSetChannelMaxTransmitPower(otInstance *aInstance, uint8_t aChannel, int8_t aMaxPower)
Set the max transmit power for a specific channel.

otError otPlatRadioSetRegion(otInstance *aInstance, uint16_t aRegionCode)
Set the region code.

otError otPlatRadioGetRegion(otInstance *aInstance, uint16_t *aRegionCode)
Get the region code.

otError otPlatRadioConfigureEnhAckProbing(otInstance *aInstance, otLinkMetrics aLinkMetrics, otShortAddress aShortAddress, const otExtAddress *aExtAddress)
Enable/disable or update Enhanced-ACK Based Probing in radio for a specific Initiator.

otError otPlatRadioAddCalibratedPower(otInstance *aInstance, uint8_t aChannel, int16_t aActualPower, const uint8_t *aRawPowerSetting, uint16_t aRawPowerSettingLength)
Add a calibrated power of the specified channel to the power calibration table.

otError otPlatRadioClearCalibratedPowers(otInstance *aInstance)
Clear all calibrated powers from the power calibration table.

otError otPlatRadioSetChannelTargetPower(otInstance *aInstance, uint8_t aChannel, int16_t aTargetPower)
Set the target power for the given channel.

otError otPlatRadioGetRawPowerSetting(otInstance *aInstance, uint8_t aChannel, uint8_t *aRawPowerSetting, uint16_t *aRawPowerSettingLength)
Get the raw power setting for the given channel.

Function Documentation

otPlatRadioGetState

```
otRadioState otPlatRadioGetState (otInstance *aInstance)
```

Get current state of the radio.

Parameters

[in]	aInstance	The OpenThread instance structure.

Is not required by OpenThread. It may be used for debugging and/or application-specific purposes.

Note

- This function may be not implemented. It does not affect OpenThread.

Returns

- Current state of the radio.

Definition at line `747` of file `include/openthread/platform/radio.h`

otPlatRadioEnable

```
otError otPlatRadioEnable (otInstance *aInstance)
```

Enable the radio.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `758` of file `include/openthread/platform/radio.h`

otPlatRadioDisable

```
otError otPlatRadioDisable (otInstance *aInstance)
```

Disable the radio.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `769` of file `include/openthread/platform/radio.h`

otPlatRadioIsEnabled

```
bool otPlatRadioIsEnabled (otInstance *aInstance)
```

Check whether radio is enabled or not.

Parameters

[in]	aInstance	The OpenThread instance structure.

Returns

- TRUE if the radio is enabled, FALSE otherwise.

Definition at line `779` of file `include/openthread/platform/radio.h`

otPlatRadioSleep

```
otError otPlatRadioSleep (otInstance *aInstance)
```

Transition the radio from Receive to Sleep (turn off the radio).

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `791` of file `include/openthread/platform/radio.h`

otPlatRadioReceive

```
otError otPlatRadioReceive (otInstance *aInstance, uint8_t aChannel)
```

Transition the radio from Sleep to Receive (turn on the radio).

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aChannel	The channel to use for receiving.

Definition at line `803` of file `include/openthread/platform/radio.h`

otPlatRadioReceiveAt

```
otError otPlatRadioReceiveAt (otInstance *aInstance, uint8_t aChannel, uint32_t aStart, uint32_t aDuration)
```

Schedule a radio reception window at a specific time and duration.

Parameters

[in]	aInstance	The radio channel on which to receive.
[in]	aChannel	The receive window start time relative to the local radio clock, see otPlatRadioGetNow . The radio receiver SHALL be on and ready to receive the first symbol of a frame's SHR at the window start time.
[in]	aStart	The receive window duration, in microseconds, as measured by the local radio clock. The radio SHOULD be turned off (or switched to TX mode if an ACK frame needs to be sent) after that duration unless it is still actively receiving a frame. In the latter case the radio SHALL be kept in reception mode until frame reception has either succeeded or failed.
N/A	aDuration	

Definition at line 824 of file include/openthread/platform/radio.h

otPlatRadioReceiveDone

```
void otPlatRadioReceiveDone (otInstance *aInstance, otRadioFrame *aFrame, otError aError)
```

The radio driver calls this method to notify OpenThread of a received frame.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aFrame	A pointer to the received frame or NULL if the receive operation failed.
[in]	aError	OT_ERROR_NONE when successfully received a frame, OT_ERROR_ABORT when reception was aborted and a frame was not received, OT_ERROR_NO_BUFS when a frame could not be received due to lack of rx buffer space.

Definition at line 836 of file include/openthread/platform/radio.h

otPlatDiagRadioReceiveDone

```
void otPlatDiagRadioReceiveDone (otInstance *aInstance, otRadioFrame *aFrame, otError aError)
```

The radio driver calls this method to notify OpenThread diagnostics module of a received frame.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aFrame	A pointer to the received frame or NULL if the receive operation failed.
[in]	aError	OT_ERROR_NONE when successfully received a frame, OT_ERROR_ABORT when reception was aborted and a frame was not received, OT_ERROR_NO_BUFS when a frame could not be received due to lack of rx buffer space.

Is used when diagnostics is enabled.

Definition at line 850 of file include/openthread/platform/radio.h

otPlatRadioGetTransmitBuffer

```
otRadioFrame * otPlatRadioGetTransmitBuffer (otInstance *aInstance)
```

Get the radio transmit frame buffer.

Parameters

[in]	aInstance	The OpenThread instance structure.

OpenThread forms the IEEE 802.15.4 frame in this buffer then calls `otPlatRadioTransmit()` to request transmission.

Returns

- A pointer to the transmit frame buffer.

Definition at line `862` of file `include/openthread/platform/radio.h`

otPlatRadioTransmit

```
otError otPlatRadioTransmit (otInstance *aInstance, otRadioFrame *aFrame)
```

Begin the transmit sequence on the radio.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aFrame	A pointer to the frame to be transmitted.

The caller must form the IEEE 802.15.4 frame in the buffer provided by `otPlatRadioGetTransmitBuffer()` before requesting transmission. The channel and transmit power are also included in the otRadioFrame structure.

The transmit sequence consists of:

1. Transitioning the radio to Transmit from one of the following states:
   - Receive if RX is on when the device is idle or OT_RADIO_CAPS_SLEEP_TO_TX is not supported
   - Sleep if RX is off when the device is idle and OT_RADIO_CAPS_SLEEP_TO_TX is supported.
2. Transmits the psdu on the given channel and at the given transmit power.

Definition at line `883` of file `include/openthread/platform/radio.h`

otPlatRadioTxStarted

```
void otPlatRadioTxStarted (otInstance *aInstance, otRadioFrame *aFrame)
```

The radio driver calls this method to notify OpenThread that the transmission has started.

Parameters

[in]	aInstance	A pointer to the OpenThread instance structure.
[in]	aFrame	A pointer to the frame that is being transmitted.

Note

- This function should be called by the same thread that executes all of the other OpenThread code. It should not be called by ISR or any other task.

Definition at line `895` of file `include/openthread/platform/radio.h`

otPlatRadioTxDone

```
void otPlatRadioTxDone (otInstance *aInstance, otRadioFrame *aFrame, otRadioFrame *aAckFrame, otError aError)
```

The radio driver calls this function to notify OpenThread that the transmit operation has completed, providing both the transmitted frame and, if applicable, the received ack frame.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aFrame	A pointer to the frame that was transmitted.
[in]	aAckFrame	A pointer to the ACK frame, NULL if no ACK was received.
[in]	aError	OT_ERROR_NONE when the frame was transmitted, OT_ERROR_NO_ACK when the frame was transmitted but no ACK was received, OT_ERROR_CHANNEL_ACCESS_FAILURE tx could not take place due to activity on the channel, OT_ERROR_ABORT when transmission was aborted for other reasons.

When radio provides `OT_RADIO_CAPS_TRANSMIT_SEC` capability, radio platform layer updates `aFrame` with the security frame counter and key index values maintained by the radio.

Definition at line `913` of file `include/openthread/platform/radio.h`

otPlatDiagRadioTransmitDone

> void otPlatDiagRadioTransmitDone (otInstance *aInstance, otRadioFrame *aFrame, otError aError)

The radio driver calls this method to notify OpenThread diagnostics module that the transmission has completed.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aFrame	A pointer to the frame that was transmitted.
[in]	aError	OT_ERROR_NONE when the frame was transmitted, OT_ERROR_CHANNEL_ACCESS_FAILURE tx could not take place due to activity on the channel, OT_ERROR_ABORT when transmission was aborted for other reasons.

Is used when diagnostics is enabled.

Definition at line `927` of file `include/openthread/platform/radio.h`

otPlatRadioGetRssi

> int8_t otPlatRadioGetRssi (otInstance *aInstance)

Get the most recent RSSI measurement.

Parameters

[in]	aInstance	The OpenThread instance structure.

Returns

- The RSSI in dBm when it is valid. 127 when RSSI is invalid.

Definition at line `937` of file `include/openthread/platform/radio.h`

otPlatRadioEnergyScan

> otError otPlatRadioEnergyScan (otInstance *aInstance, uint8_t aScanChannel, uint16_t aScanDuration)

Begin the energy scan sequence on the radio.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aScanChannel	The channel to perform the energy scan on.
[in]	aScanDuration	The duration, in milliseconds, for the channel to be scanned.

Is used when radio provides OT_RADIO_CAPS_ENERGY_SCAN capability.

Definition at line `953` of file `include/openthread/platform/radio.h`

otPlatRadioEnergyScanDone

```
void otPlatRadioEnergyScanDone (otInstance *aInstance, int8_t aEnergyScanMaxRssi)
```

The radio driver calls this method to notify OpenThread that the energy scan is complete.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aEnergyScanMaxRssi	The maximum RSSI encountered on the scanned channel.

Is used when radio provides OT_RADIO_CAPS_ENERGY_SCAN capability.

Definition at line `964` of file `include/openthread/platform/radio.h`

otPlatRadioEnableSrcMatch

```
void otPlatRadioEnableSrcMatch (otInstance *aInstance, bool aEnable)
```

Enable/Disable source address match feature.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aEnable	Enable/disable source address match feature.

The source address match feature controls how the radio layer decides the "frame pending" bit for acks sent in response to data request commands from children.

If disabled, the radio layer must set the "frame pending" on all acks to data request commands.

If enabled, the radio layer uses the source address match table to determine whether to set or clear the "frame pending" bit in an ack to a data request command.

The source address match table provides the list of children for which there is a pending frame. Either a short address or an extended/long address can be added to the source address match table.

Definition at line `984` of file `include/openthread/platform/radio.h`

otPlatRadioAddSrcMatchShortEntry

```
otError otPlatRadioAddSrcMatchShortEntry (otInstance *aInstance, otShortAddress aShortAddress)
```

Add a short address to the source address match table.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aShortAddress	The short address to be added.

Definition at line 996 of file include/openthread/platform/radio.h

otPlatRadioAddSrcMatchExtEntry

> otError otPlatRadioAddSrcMatchExtEntry (otInstance *aInstance, const otExtAddress *aExtAddress)

Add an extended address to the source address match table.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aExtAddress	The extended address to be added stored in little-endian byte order.

Definition at line 1008 of file include/openthread/platform/radio.h

otPlatRadioClearSrcMatchShortEntry

> otError otPlatRadioClearSrcMatchShortEntry (otInstance *aInstance, otShortAddress aShortAddress)

Remove a short address from the source address match table.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aShortAddress	The short address to be removed.

Definition at line 1020 of file include/openthread/platform/radio.h

otPlatRadioClearSrcMatchExtEntry

> otError otPlatRadioClearSrcMatchExtEntry (otInstance *aInstance, const otExtAddress *aExtAddress)

Remove an extended address from the source address match table.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aExtAddress	The extended address to be removed stored in little-endian byte order.

Definition at line 1032 of file include/openthread/platform/radio.h

otPlatRadioClearSrcMatchShortEntries

> void otPlatRadioClearSrcMatchShortEntries (otInstance *aInstance)

Clear all short addresses from the source address match table.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line 1040 of file include/openthread/platform/radio.h

otPlatRadioClearSrcMatchExtEntries

```
void otPlatRadioClearSrcMatchExtEntries (otInstance *aInstance)
```

Clear all the extended/long addresses from source address match table.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line 1048 of file include/openthread/platform/radio.h

otPlatRadioGetSupportedChannelMask

```
uint32_t otPlatRadioGetSupportedChannelMask (otInstance *aInstance)
```

Get the radio supported channel mask that the device is allowed to be on.

Parameters

[in]	aInstance	The OpenThread instance structure.

Returns

- The radio supported channel mask.

Definition at line 1058 of file include/openthread/platform/radio.h

otPlatRadioGetPreferredChannelMask

```
uint32_t otPlatRadioGetPreferredChannelMask (otInstance *aInstance)
```

Gets the radio preferred channel mask that the device prefers to form on.

Parameters

[in]	aInstance	The OpenThread instance structure.

Returns

- The radio preferred channel mask.

Definition at line 1068 of file include/openthread/platform/radio.h

otPlatRadioSetCoexEnabled

```
otError otPlatRadioSetCoexEnabled (otInstance *aInstance, bool aEnabled)
```

Enable the radio coex.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aEnabled	TRUE to enable the radio coex, FALSE otherwise.

Is used when feature OPENTHREAD_CONFIG_PLATFORM_RADIO_COEX_ENABLE is enabled.

Definition at line 1082 of file include/openthread/platform/radio.h

otPlatRadioIsCoexEnabled

> bool otPlatRadioIsCoexEnabled (otInstance *aInstance)

Check whether radio coex is enabled or not.

Parameters

[in]	aInstance	The OpenThread instance structure.

Is used when feature OPENTHREAD_CONFIG_PLATFORM_RADIO_COEX_ENABLE is enabled.

Returns

- TRUE if the radio coex is enabled, FALSE otherwise.

Definition at line 1094 of file include/openthread/platform/radio.h

otPlatRadioGetCoexMetrics

> otError otPlatRadioGetCoexMetrics (otInstance *aInstance, otRadioCoexMetrics *aCoexMetrics)

Get the radio coexistence metrics.

Parameters

[in]	aInstance	The OpenThread instance structure.
[out]	aCoexMetrics	A pointer to the coexistence metrics structure.

Is used when feature OPENTHREAD_CONFIG_PLATFORM_RADIO_COEX_ENABLE is enabled.

Definition at line 1107 of file include/openthread/platform/radio.h

otPlatRadioEnableCsl

> otError otPlatRadioEnableCsl (otInstance *aInstance, uint32_t aCslPeriod, otShortAddress aShortAddr, const otExtAddress *aExtAddr)

Enable or disable CSL receiver.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aCslPeriod	CSL period, 0 for disabling CSL. CSL period is in unit of 10 symbols.
[in]	aShortAddr	The short source address of CSL receiver's peer.
[in]	aExtAddr	The extended source address of CSL receiver's peer.

Note

Platforms should use CSL peer addresses to include CSL IE when generating enhanced acks.

Definition at line 1124 of file include/openthread/platform/radio.h

otPlatRadioUpdateCslSampleTime

```
void otPlatRadioUpdateCslSampleTime (otInstance *aInstance, uint32_t aCslSampleTime)
```

Update CSL sample time in radio driver.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aCslSampleTime	The next sample time, in microseconds. It is the time when the first symbol of the MHR of the frame is expected.

Sample time is stored in radio driver as a copy to calculate phase when sending ACK with CSL IE. The CSL sample (window) of the CSL receiver extends before and after the sample time. The CSL sample time marks a timestamp in the CSL sample window when a frame should be received in "ideal conditions" if there would be no inaccuracy/clock-drift.

Definition at line 1143 of file include/openthread/platform/radio.h

otPlatRadioGetCslAccuracy

```
uint8_t otPlatRadioGetCslAccuracy (otInstance *aInstance)
```

Get the current estimated worst case accuracy (maximum ± deviation from the nominal frequency) of the local radio clock in units of PPM.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

This is the clock used to schedule CSL operations.

Note

- Implementations MAY estimate this value based on current operating conditions (e.g. temperature).

In case the implementation does not estimate the current value but returns a fixed value, this value MUST be the worst-case accuracy over all possible foreseen operating conditions (temperature, pressure, etc) of the implementation.

Returns

- The current CSL rx/tx scheduling drift, in PPM.

Definition at line 1163 of file include/openthread/platform/radio.h

otPlatRadioGetCslUncertainty

```
uint8_t otPlatRadioGetCslUncertainty (otInstance *aInstance)
```

The fixed uncertainty (i.e.

Parameters

[in]	aInstance	A pointer to an OpenThread instance.

random jitter) of the arrival time of CSL transmissions received by this device in units of 10 microseconds.

This designates the worst case constant positive or negative deviation of the actual arrival time of a transmission from the transmission time calculated relative to the local radio clock independent of elapsed time. In addition to uncertainty accumulated over elapsed time, the CSL channel sample ("RX window") must be extended by twice this deviation such that an actual transmission is guaranteed to be detected by the local receiver in the presence of random arrival time jitter.

Returns

- The CSL Uncertainty in units of 10 us.

Definition at line `1182` of file `include/openthread/platform/radio.h`

otPlatRadioSetChannelMaxTransmitPower

> otError otPlatRadioSetChannelMaxTransmitPower (otInstance *aInstance, uint8_t aChannel, int8_t aMaxPower)

Set the max transmit power for a specific channel.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aChannel	The radio channel.
[in]	aMaxPower	The max power in dBm, passing OT_RADIO_RSSI_INVALID will disable this channel.

Definition at line `1197` of file `include/openthread/platform/radio.h`

otPlatRadioSetRegion

> otError otPlatRadioSetRegion (otInstance *aInstance, uint16_t aRegionCode)

Set the region code.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aRegionCode	The radio region code. The `aRegionCode >> 8` is first ascii char and the `aRegionCode & 0xff` is the second ascii char.

The radio region format is the 2-bytes ascii representation of the ISO 3166 alpha-2 code.

Definition at line `1214` of file `include/openthread/platform/radio.h`

otPlatRadioGetRegion

> otError otPlatRadioGetRegion (otInstance *aInstance, uint16_t *aRegionCode)

Get the region code.

Parameters

[in]	aInstance	The OpenThread instance structure.
[out]	aRegionCode	The radio region.

The radio region format is the 2-bytes ascii representation of the ISO 3166 alpha-2 code.

Definition at line 1231 of file include/openthread/platform/radio.h

otPlatRadioConfigureEnhAckProbing

> otError otPlatRadioConfigureEnhAckProbing (otInstance *aInstance, otLinkMetrics aLinkMetrics, otShortAddress aShortAddress, const otExtAddress *aExtAddress)

Enable/disable or update Enhanced-ACK Based Probing in radio for a specific Initiator.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aLinkMetrics	This parameter specifies what metrics to query. Per spec 4.11.3.4.4.6, at most 2 metrics can be specified. The probing would be disabled if `aLinkMetrics` is bitwise 0.
[in]	aShortAddress	The short address of the Probing Initiator.
[in]	aExtAddress	The extended source address of the Probing Initiator. aExtAddr MUST NOT be NULL.

After Enhanced-ACK Based Probing is configured by a specific Probing Initiator, the Enhanced-ACK sent to that node should include Vendor-Specific IE containing Link Metrics data. This method informs the radio to start/stop to collect Link Metrics data and include Vendor-Specific IE that containing the data in Enhanced-ACK sent to that Probing Initiator.

Definition at line 1254 of file include/openthread/platform/radio.h

otPlatRadioAddCalibratedPower

> otError otPlatRadioAddCalibratedPower (otInstance *aInstance, uint8_t aChannel, int16_t aActualPower, const uint8_t *aRawPowerSetting, uint16_t aRawPowerSettingLength)

Add a calibrated power of the specified channel to the power calibration table.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aChannel	The radio channel.
[in]	aActualPower	The actual power in 0.01dBm.
[in]	aRawPowerSetting	A pointer to the raw power setting byte array.
[in]	aRawPowerSettingLength	The length of the aRawPowerSetting.

Note

- This API is an optional radio platform API. It's up to the platform layer to implement it.

The aActualPower is the actual measured output power when the parameters of the radio hardware modules are set to the aRawPowerSetting.

The raw power setting is an opaque byte array. OpenThread doesn't define the format of the raw power setting. Its format is radio hardware related and it should be defined by the developers in the platform radio driver. For example, if the radio hardware contains both the radio chip and the FEM chip, the raw power setting can be a combination of the radio power register and the FEM gain value.

Definition at line 1285 of file include/openthread/platform/radio.h

otPlatRadioClearCalibratedPowers

> otError otPlatRadioClearCalibratedPowers (otInstance *aInstance)

Clear all calibrated powers from the power calibration table.

Parameters

[in]	aInstance	The OpenThread instance structure.

Note

- This API is an optional radio platform API. It's up to the platform layer to implement it.

Definition at line `1302` of file `include/openthread/platform/radio.h`

otPlatRadioSetChannelTargetPower

> otError otPlatRadioSetChannelTargetPower (otInstance *aInstance, uint8_t aChannel, int16_t aTargetPower)

Set the target power for the given channel.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aChannel	The radio channel.
[in]	aTargetPower	The target power in 0.01dBm. Passing `INT16_MAX` will disable this channel to use the target power.

Note

- This API is an optional radio platform API. It's up to the platform layer to implement it. If this API is implemented, the function `otPlatRadioSetTransmitPower()` should be disabled.

The radio driver should set the actual output power to be less than or equal to the target power and as close as possible to the target power.

Definition at line `1323` of file `include/openthread/platform/radio.h`

otPlatRadioGetRawPowerSetting

> otError otPlatRadioGetRawPowerSetting (otInstance *aInstance, uint8_t aChannel, uint8_t *aRawPowerSetting, uint16_t *aRawPowerSettingLength)

Get the raw power setting for the given channel.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aChannel	The radio channel.
[out]	aRawPowerSetting	A pointer to the raw power setting byte array.
[inout]	aRawPowerSettingLength	On input, a pointer to the size of `aRawPowerSetting`. On output, a pointer to the length of the raw power setting data.

Note

- OpenThread `src/core/utils` implements a default implementation of the API `otPlatRadioAddCalibratedPower()`, `otPlatRadioClearCalibratedPowers()` and `otPlatRadioSetChannelTargetPower()`. This API is provided by the default implementation to get the raw power setting for the given channel. If the platform doesn't use the default implementation, it can ignore this API.

Platform radio layer should parse the raw power setting based on the radio layer defined format and set the parameters of each radio hardware module.

Definition at line 1348 of file include/openthread/platform/radio.h

Radio Extension

Radio Extension

This module includes the Silicon Labs extension to the openthread platform radio interface.

The functions in this modules provide an API that can be called from SoC or host based openthread applications.

Note

- Many of the functions defined in this module are wrappers on top of the Silicon Labs RAIL API. For additional information on the RAII API please refer to the `Silicon Labs RAIL API Reference Guide`. Those functions that are wrappers to RAIL functions include a reference to the underlying RAIL function.

Enumerations

enum otPlatRadioExtensionCoexEvent_t {

OT_PLAT_RADIO_EXTENSION_COEX_EVENT_LO_PRI_REQUESTED
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_HI_PRI_REQUESTED
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_LO_PRI_DENIED
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_HI_PRI_DENIED
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_LO_PRI_TX_ABORTED
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_HI_PRI_TX_ABORTED
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_COUNT

}

This enumeration defines the coex event counters and can be used as an index into the `aCoexCounters` table returned in a call to otPlatRadioExtensionGetCoexCounters.

Functions

otError otPlatRadioExtensionGetTxAntennaMode(uint8_t *aMode)
Get the antenna diversity transmit antenna mode.

otError otPlatRadioExtensionSetTxAntennaMode(uint8_t aMode)
Set the antenna diversity transmit antenna mode.

otError otPlatRadioExtensionGetRxAntennaMode(uint8_t *aMode)
Get the antenna diversity receive antenna mode.

otError otPlatRadioExtensionSetRxAntennaMode(uint8_t aMode)
Set the antenna diversity receive antenna mode.

otError otPlatRadioExtensionGetActivePhy(uint8_t *aActivePhy)
Get the antenna diversity active phy state.

otError otPlatRadioExtensionGetDpState(uint8_t *aDpPulse)
Get the coexistence directional priority state and pulse width.

otError otPlatRadioExtensionSetDpState(uint8_t aDpPulse)
Set the coexistence directional priority state and pulse width.

otError otPlatRadioExtensionGetGpioInputOverride(uint8_t aGpioIndex, bool *aEnabled)
Get the override input value of a GPIO.

otError otPlatRadioExtensionSetGpioInputOverride(uint8_t aGpioIndex, bool aEnabled)
Set the override input value of a GPIO.

otError otPlatRadioExtensionGetActiveRadio(uint8_t *aActivePhy)
Get the coexistence active phy state.

otError otPlatRadioExtensionGetPhySelectTimeout(uint8_t *aTimeout)
Get the coexistence phy select state and timeout.

otError otPlatRadioExtensionSetPhySelectTimeout(uint8_t aTimeout)
Set the coexistence phy select state and timeout.

otError otPlatRadioExtensionGetCoexOptions(uint32_t *aPtaOptions)
Get the coexistence bitmask of features.

otError otPlatRadioExtensionSetCoexOptions(uint32_t aPtaOptions)
Set the coexistence bitmask of features.

otError otPlatRadioExtensionGetCoexConstantOptions(uint32_t *aPtaOptions)
Get the coexistence bitmask of constant PTA features that can not be modified using public APIs.

otError otPlatRadioExtensionIsCoexEnabled(bool *aPtaState)
Get the coexistence enabled status.

otError otPlatRadioExtensionSetCoexEnable(bool aPtaState)
Set the coexistence enabled status.

otError otPlatRadioExtensionGetRequestPwmArgs(uint8_t *aPwmReq, uint8_t *aPwmDutyCycle, uint8_t *aPwmPeriodHalfMs)
Get the coexistence PWM configuration.

otError otPlatRadioExtensionSetRequestPwmArgs(uint8_t aPwmReq, uint8_t aPwmDutyCycle, uint8_t aPwmPeriodHalfMs)
Set the coexistence PWM configuration.

otError otPlatRadioExtensionClearCoexCounters(void)
Clear the coexistence counters.

otError otPlatRadioExtensionGetCoexCounters(uint8_t aNumEntries, uint32_t aCoexCounters[])
Get the coexistence counters.

otError otPlatRadioExtensionSetRadioHoldoff(bool aEnabled)
Set the coexistence radio holdoff status.

otError otPlatRadioExtensionGetRadioCounters(efr32RadioCounters *aCounters)
Get RAIL debug counter values.

otError otPlatRadioExtensionClearRadioCounters(void)
Clear the RAIL debug counters.

Enumeration Documentation

otPlatRadioExtensionCoexEvent_t

> otPlatRadioExtensionCoexEvent_t

This enumeration defines the coex event counters and can be used as an index into the `aCoexCounters` table returned in a call to otPlatRadioExtensionGetCoexCounters.

Enumerator	
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_LO_PRI_REQUESTED	Low priority request initiated.

OT_PLAT_RADIO_EXTENSION_COEX_EVENT_HI_PRI_REQUESTED	High priority request initiated.
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_LO_PRI_DENIED	Low priority request denied.
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_HI_PRI_DENIED	High priority request denied.
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_LO_PRI_TX_ABORTED	Low priority transmission aborted mid packet.
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_HI_PRI_TX_ABORTED	High priority transmission aborted mid packet.
OT_PLAT_RADIO_EXTENSION_COEX_EVENT_COUNT	Number of coexistence events.

Definition at line `633` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

Function Documentation

otPlatRadioExtensionGetTxAntennaMode

```
otError otPlatRadioExtensionGetTxAntennaMode (uint8_t *aMode)
```

Get the antenna diversity transmit antenna mode.

Parameters

[out]	aMode	A pointer to the location where the current transmit antenna mode will be returned. Antenna modes are defined by the RAIL `sl_rail_util_antenna_mode_t` enumeration.

Requires the `ot_ant_div` component.

See Also

- RAIL API: **sl_rail_util_ant_div_get_tx_antenna_mode()**

Returns

- Error code indicating success of the function call.

Definition at line `97` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionSetTxAntennaMode

```
otError otPlatRadioExtensionSetTxAntennaMode (uint8_t aMode)
```

Set the antenna diversity transmit antenna mode.

Parameters

[in]	aMode	The antenna mode to use for transmit. Antenna modes are defined by the RAIL `sl_rail_util_antenna_mode_t` enumeration.

Requires the `ot_ant_div` component.

See Also

- RAIL API: **sl_rail_util_ant_div_set_tx_antenna_mode()**

Returns

- Error code indicating success of the function call.

Definition at line 119 of file /mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h

otPlatRadioExtensionGetRxAntennaMode

> otError otPlatRadioExtensionGetRxAntennaMode (uint8_t *aMode)

Get the antenna diversity receive antenna mode.

Parameters

[out]	aMode	A pointer to the location where the current receive antenna mode will be returned. Antenna modes are defined by the RAIL sl_rail_util_antenna_mode_t enumeration.

Requires the ot_ant_div component.

See Also

- RAIL API: **sl_rail_util_ant_div_get_rx_antenna_mode()**

Returns

- Error code indicating success of the function call.

Definition at line 142 of file /mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h

otPlatRadioExtensionSetRxAntennaMode

> otError otPlatRadioExtensionSetRxAntennaMode (uint8_t aMode)

Set the antenna diversity receive antenna mode.

Parameters

[in]	aMode	The antenna mode to use for receive. Antenna modes are defined by the RAIL sl_rail_util_antenna_mode_t enumeration.

Requires the ot_ant_div component.

See Also

- RAIL API: **sl_rail_util_ant_div_set_rx_antenna_mode()**

Returns

- Error code indicating success of the function call.

Definition at line 164 of file /mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h

otPlatRadioExtensionGetActivePhy

> otError otPlatRadioExtensionGetActivePhy (uint8_t *aActivePhy)

Get the antenna diversity active phy state.

Parameters

[out]	aActivePhy	A pointer to the location where the current phy state will be returned. Phy states are defined by the RAIL `sl_rail_util_ieee802154_radio_config_t` enumeration.

Requires the `ot_ant_div` component.

See Also

- RAIL API: **sl_rail_util_ieee802154_get_active_radio_config()**

Returns

- Error code indicating success of the function call.

Definition at line `187` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionGetDpState

```
otError otPlatRadioExtensionGetDpState (uint8_t *aDpPulse)
```

Get the coexistence directional priority state and pulse width.

Parameters

[out]	aDpPulse	A pointer to the location where the current directional priority state will be returned. If `aDpPulse` is 0 then directional priority is disabled. If `aDpPulse` is not 0 then directional priority is enabled and the value is the pulse width in microseconds.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_get_directional_priority_pulse_width()**

Returns

- Error code indicating success of the function call.

Definition at line `211` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionSetDpState

```
otError otPlatRadioExtensionSetDpState (uint8_t aDpPulse)
```

Set the coexistence directional priority state and pulse width.

Parameters

[in]	aDpPulse	The directional priority state to set. If `aDpPulse` is 0 then directional priority will be disabled. If `aDpPulse` is not 0 then directional priority will be enabled and the value will be the pulse width to use in microseconds.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_set_directional_priority_pulse_width()**

Returns

Error code indicating success of the function call.

Definition at line `234` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionGetGpioInputOverride

> otError otPlatRadioExtensionGetGpioInputOverride (uint8_t aGpioIndex, bool *aEnabled)

Get the override input value of a GPIO.

Parameters

[in]	aGpioIndex	The GPIO index • 0x00 = Radio Holdoff GPIO index • 0x01 = Request GPIO index • 0x02 = Grant GPIO index • 0x03 = PHY Select index
[out]	aEnabled	A pointer to the location where the boolean override input value will be returned. A TRUE value indicating the override input value is enabled, FALSE disabled. The return is inverted if the selected GPIO is active low.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_get_gpio_input_override()**

Returns

- Error code indicating success of the function call.

Definition at line `262` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionSetGpioInputOverride

> otError otPlatRadioExtensionSetGpioInputOverride (uint8_t aGpioIndex, bool aEnabled)

Set the override input value of a GPIO.

Parameters

[in]	aGpioIndex	The GPIO index • 0x00 = Radio Holdoff GPIO index • 0x01 = Request GPIO index • 0x02 = Grant GPIO index • 0x03 = PHY Select index
[in]	aEnabled	The boolean override input value. A TRUE value indicating the override input value is enabled, FALSE disabled.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_set_gpio_input_override()**

Returns

- Error code indicating success of the function call.

Definition at line `289` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionGetActiveRadio

```
otError otPlatRadioExtensionGetActiveRadio (uint8_t *aActivePhy)
```

Get the coexistence active phy state.

Parameters

[out]	aActivePhy	A pointer to the location where the current phy state will be returned. Phy states are defined by the RAIL `sl_rail_util_ieee802154_radio_config_t` enumeration.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_ieee802154_get_active_radio_config()**

Returns

- Error code indicating success of the function call.

Definition at line `312` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionGetPhySelectTimeout

```
otError otPlatRadioExtensionGetPhySelectTimeout (uint8_t *aTimeout)
```

Get the coexistence phy select state and timeout.

Parameters

[out]	aTimeout	A pointer to the location where the current phy select state will be returned. If `aTimeout` is 0 then phy select is disabled. If `aTimeout` is not 0 then phy select is enabled and the value is the timeout in milliseconds.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_get_phy_select_timeout()**

Returns

- Error code indicating success of the function call.

Definition at line `336` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionSetPhySelectTimeout

```
otError otPlatRadioExtensionSetPhySelectTimeout (uint8_t aTimeout)
```

Set the coexistence phy select state and timeout.

Parameters

[in]	aTimeout	The phy select state to set. If `aTimeout` is 0 then phy select will be disabled. If `aTimeout` is not 0 then phy select will be enabled and the value will be the timeout to use in milliseconds.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_set_phy_select_timeout()**

Returns

- Error code indicating success of the function call.

Definition at line `359` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionGetCoexOptions

> otError otPlatRadioExtensionGetCoexOptions (uint32_t *aPtaOptions)

Get the coexistence bitmask of features.

Parameters

[out]	aPtaOptions	A pointer to the location where the coexistence feature bitmask will be returned. The feature bitmask is defined by the set of macros making up the RAIL `sl_rail_util_coex_options_t` type.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_get_options()**

Returns

- Error code indicating success of the function call.

Definition at line `382` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionSetCoexOptions

> otError otPlatRadioExtensionSetCoexOptions (uint32_t aPtaOptions)

Set the coexistence bitmask of features.

Parameters

[in]	aPtaOptions	The coexistence feature bitmask to set. The feature bitmask is defined by the set of macros making up the RAIL `sl_rail_util_coex_options_t` type.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_set_options()**

Returns

- Error code indicating success of the function call.

Definition at line `405` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionGetCoexConstantOptions

```
otError otPlatRadioExtensionGetCoexConstantOptions (uint32_t *aPtaOptions)
```

Get the coexistence bitmask of constant PTA features that can not be modified using public APIs.

Parameters

[out]	aPtaOptions	A pointer to the location where the coexistence constant PTA feature bitmask will be returned. The feature bitmask is defined by the set of macros making up the RAIL `sl_rail_util_coex_options_t` type.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_get_constant_options()**

Returns

- Error code indicating success of the function call.

Definition at line `428` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionIsCoexEnabled

```
otError otPlatRadioExtensionIsCoexEnabled (bool *aPtaState)
```

Get the coexistence enabled status.

Parameters

[out]	aPtaState	A pointer to the location where the coexistence enabled status will be returned.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_is_enabled()**

Returns

- Error code indicating success of the function call.

Definition at line `449` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionSetCoexEnable

```
otError otPlatRadioExtensionSetCoexEnable (bool aPtaState)
```

Set the coexistence enabled status.

Parameters

[in]	aPtaState	The coexistence enabled status.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_set_enable()**

Returns

- Error code indicating success of the function call.

Definition at line `469` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionGetRequestPwmArgs

> otError otPlatRadioExtensionGetRequestPwmArgs (uint8_t *aPwmReq, uint8_t *aPwmDutyCycle, uint8_t *aPwmPeriodHalfMs)

Get the coexistence PWM configuration.

Parameters

[out]	aPwmReq	A pointer to the location where the coexistence PWM request is returned. The value is defined as a bitmap using shift values from the RAIL `COEX_Req_t` enumeration.
[out]	aPwmDutyCycle	A pointer to the location where the coexistence PWM duty cycle value is returned.
[out]	aPwmPeriodHalfMs	A pointer to the location where the coexistence PWM period half MS value is returned.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_get_request_pwm_args()**

Returns

- Error code indicating success of the function call.

Definition at line `497` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionSetRequestPwmArgs

> otError otPlatRadioExtensionSetRequestPwmArgs (uint8_t aPwmReq, uint8_t aPwmDutyCycle, uint8_t aPwmPeriodHalfMs)

Set the coexistence PWM configuration.

Parameters

[in]	aPwmReq	The coexistence PWM request. The value is defined as a bitmap using shift values from the RAIL `COEX_Req_t` enumeration.
[in]	aPwmDutyCycle	The coexistence PWM duty cycle.
[in]	aPwmPeriodHalfMs	The coexistencec PWM period half MS.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_set_request_pwm()**

Returns

- Error code indicating success of the function call.

Definition at line `521` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionClearCoexCounters

> otError otPlatRadioExtensionClearCoexCounters (void)

Clear the coexistence counters.

Parameters

N/A		

Requires the `ot_coex` component.

Returns

- Error code indicating success of the function call.

Definition at line `537` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionGetCoexCounters

> otError otPlatRadioExtensionGetCoexCounters (uint8_t aNumEntries, uint32_t aCoexCounters[])

Get the coexistence counters.

Parameters

[in]	aNumEntries	The number of entries in `aCoexCounters` array where counters will be returned.
[out]	aCoexCounters	A pointer to an array where the coexistence counters will be returned. See otPlatRadioExtensionCoexEvent_t which defines what coexistence counter each array element stores.

Requires the `ot_coex` component.

Returns

- Error code indicating success of the function call.

Definition at line `562` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionSetRadioHoldoff

> otError otPlatRadioExtensionSetRadioHoldoff (bool aEnabled)

Set the coexistence radio holdoff status.

Parameters

[in]	aEnabled	The coexistence radio holdoff status.

Requires the `ot_coex` component.

See Also

- RAIL API: **sl_rail_util_coex_set_radio_holdoff()**

Returns

- Error code indicating success of the function call.

Definition at line `582` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionGetRadioCounters

> otError otPlatRadioExtensionGetRadioCounters (efr32RadioCounters *aCounters)

Get RAIL debug counter values.

Parameters

[out]	aCounters	Pointer to struct to store counter values.

Requires the `ot_efr32_custom_cli` component.

Returns

- Error code indicating success of the function call.

Definition at line `666` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

otPlatRadioExtensionClearRadioCounters

> otError otPlatRadioExtensionClearRadioCounters (void)

Clear the RAIL debug counters.

Parameters

N/A		

Requires the `ot_efr32_custom_cli` component.

Returns

- Error code indicating success of the function call.

Definition at line `681` of file `/mnt/raid/workspaces/ws.QBNpfqGmL/overlay/gsdk/protocol/openthread/platform-abstraction/include/radio_extension.h`

Settings

Settings

This module includes the platform abstraction for non-volatile storage of settings.

Enumerations

enum @21 {

OT_SETTINGS_KEY_ACTIVE_DATASET = 0x0001
OT_SETTINGS_KEY_PENDING_DATASET = 0x0002
OT_SETTINGS_KEY_NETWORK_INFO = 0x0003
OT_SETTINGS_KEY_PARENT_INFO = 0x0004
OT_SETTINGS_KEY_CHILD_INFO = 0x0005
OT_SETTINGS_KEY_SLAAC_IID_SECRET_KEY = 0x0007
OT_SETTINGS_KEY_DAD_INFO = 0x0008
OT_SETTINGS_KEY_SRP_ECDSA_KEY = 0x000b
OT_SETTINGS_KEY_SRP_CLIENT_INFO = 0x000c
OT_SETTINGS_KEY_SRP_SERVER_INFO = 0x000d
OT_SETTINGS_KEY_BR_ULA_PREFIX = 0x000f
OT_SETTINGS_KEY_BR_ON_LINK_PREFIXES = 0x0010
OT_SETTINGS_KEY_BORDER_AGENT_ID = 0x0011
OT_SETTINGS_KEY_VENDOR_RESERVED_MIN = 0x8000
OT_SETTINGS_KEY_VENDOR_RESERVED_MAX = 0xffff

}

Defines the keys of settings.

Functions

void otPlatSettingsInit(otInstance *aInstance, const uint16_t *aSensitiveKeys, uint16_t aSensitiveKeysLength)
Performs any initialization for the settings subsystem, if necessary.

void otPlatSettingsDeinit(otInstance *aInstance)
Performs any de-initialization for the settings subsystem, if necessary.

otError otPlatSettingsGet(otInstance *aInstance, uint16_t aKey, int aIndex, uint8_t *aValue, uint16_t *aValueLength)
Fetches the value of a setting.

otError otPlatSettingsSet(otInstance *aInstance, uint16_t aKey, const uint8_t *aValue, uint16_t aValueLength)
Sets or replaces the value of a setting.

otError otPlatSettingsAdd(otInstance *aInstance, uint16_t aKey, const uint8_t *aValue, uint16_t aValueLength)
Adds a value to a setting.

otError otPlatSettingsDelete(otInstance *aInstance, uint16_t aKey, int aIndex)
Removes a setting from the setting store.

void otPlatSettingsWipe(otInstance *aInstance)
Removes all settings from the setting store.

Enumeration Documentation

@21

@21

Defines the keys of settings.

Note: When adding a new settings key, if the settings corresponding to the key contains security sensitive information, the developer MUST add the key to the array `aSensitiveKeys` which is passed in `otPlatSettingsInit()`.

Enumerator	
OT_SETTINGS_KEY_ACTIVE_DATASET	Active Operational Dataset.
OT_SETTINGS_KEY_PENDING_DATASET	Pending Operational Dataset.
OT_SETTINGS_KEY_NETWORK_INFO	Thread network information.
OT_SETTINGS_KEY_PARENT_INFO	Parent information.
OT_SETTINGS_KEY_CHILD_INFO	Child information.
OT_SETTINGS_KEY_SLAAC_IID_SECRET_KEY	SLAAC key to generate semantically opaque IID.
OT_SETTINGS_KEY_DAD_INFO	Duplicate Address Detection (DAD) information.
OT_SETTINGS_KEY_SRP_ECDSA_KEY	SRP client ECDSA public/private key pair.
OT_SETTINGS_KEY_SRP_CLIENT_INFO	The SRP client info (selected SRP server address).
OT_SETTINGS_KEY_SRP_SERVER_INFO	The SRP server info (UDP port).
OT_SETTINGS_KEY_BR_ULA_PREFIX	BR ULA prefix.
OT_SETTINGS_KEY_BR_ON_LINK_PREFIXES	BR local on-link prefixes.
OT_SETTINGS_KEY_BORDER_AGENT_ID	Unique Border Agent/Router ID.
OT_SETTINGS_KEY_VENDOR_RESERVED_MIN	
OT_SETTINGS_KEY_VENDOR_RESERVED_MAX	

Definition at line `62` of file `include/openthread/platform/settings.h`

Function Documentation

otPlatSettingsInit

```
void otPlatSettingsInit (otInstance *aInstance, const uint16_t *aSensitiveKeys, uint16_t aSensitiveKeysLength)
```

Performs any initialization for the settings subsystem, if necessary.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aSensitiveKeys	A pointer to an array containing the list of sensitive keys. May be NULL only if `aSensitiveKeysLength` is 0, which means that there is no sensitive keys.
[in]	aSensitiveKeysLength	The number of entries in the `aSensitiveKeys` array.

Also sets the sensitive keys that should be stored in the secure area.

Note that the memory pointed by `aSensitiveKeys` MUST not be released before `aInstance` is destroyed.

Definition at line `103` of file `include/openthread/platform/settings.h`

otPlatSettingsDeinit

```
void otPlatSettingsDeinit (otInstance *aInstance)
```

Performs any de-initialization for the settings subsystem, if necessary.

Parameters

[in]	aInstance	The OpenThread instance structure.

Definition at line `111` of file `include/openthread/platform/settings.h`

otPlatSettingsGet

> otError otPlatSettingsGet (otInstance *aInstance, uint16_t aKey, int aIndex, uint8_t *aValue, uint16_t *aValueLength)

Fetches the value of a setting.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aKey	The key associated with the requested setting.
[in]	aIndex	The index of the specific item to get.
[out]	aValue	A pointer to where the value of the setting should be written. May be set to NULL if just testing for the presence or length of a setting.
[inout]	aValueLength	A pointer to the length of the value. When called, this pointer should point to an integer containing the maximum value size that can be written to `aValue`. At return, the actual length of the setting is written. This may be set to NULL if performing a presence check.

Fetches the value of the setting identified by `aKey` and write it to the memory pointed to by aValue. It then writes the length to the integer pointed to by `aValueLength`. The initial value of `aValueLength` is the maximum number of bytes to be written to `aValue`.

Can be used to check for the existence of a key without fetching the value by setting `aValue` and `aValueLength` to NULL. You can also check the length of the setting without fetching it by setting only aValue to NULL.

Note that the underlying storage implementation is not required to maintain the order of settings with multiple values. The order of such values MAY change after ANY write operation to the store.

Definition at line `147` of file `include/openthread/platform/settings.h`

otPlatSettingsSet

> otError otPlatSettingsSet (otInstance *aInstance, uint16_t aKey, const uint8_t *aValue, uint16_t aValueLength)

Sets or replaces the value of a setting.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aKey	The key associated with the setting to change.
[in]	aValue	A pointer to where the new value of the setting should be read from. MUST NOT be NULL if `aValueLength` is non-zero.
[in]	aValueLength	The length of the data pointed to by aValue. May be zero.

Sets or replaces the value of a setting identified by `aKey`.

Calling this function successfully may cause unrelated settings with multiple values to be reordered.

OpenThread stack guarantees to use `otPlatSettingsSet()` method for a `aKey` that was either previously set using `otPlatSettingsSet()` (i.e., contains a single value) or is empty and/or fully deleted (contains no value).

Platform layer can rely and use this fact for optimizing its implementation.

Definition at line `176` of file `include/openthread/platform/settings.h`

otPlatSettingsAdd

```
otError otPlatSettingsAdd (otInstance *aInstance, uint16_t aKey, const uint8_t *aValue, uint16_t aValueLength)
```

Adds a value to a setting.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aKey	The key associated with the setting to change.
[in]	aValue	A pointer to where the new value of the setting should be read from. MUST NOT be NULL if `aValueLength` is non-zero.
[in]	aValueLength	The length of the data pointed to by `aValue` . May be zero.

Adds the value to a setting identified by `aKey` , without replacing any existing values.

Note that the underlying implementation is not required to maintain the order of the items associated with a specific key. The added value may be added to the end, the beginning, or even somewhere in the middle. The order of any pre-existing values may also change.

Calling this function successfully may cause unrelated settings with multiple values to be reordered.

OpenThread stack guarantees to use `otPlatSettingsAdd()` method for a `aKey` that was either previously managed by `otPlatSettingsAdd()` (i.e., contains one or more items) or is empty and/or fully deleted (contains no value).

Platform layer can rely and use this fact for optimizing its implementation.

Definition at line `212` of file `include/openthread/platform/settings.h`

otPlatSettingsDelete

```
otError otPlatSettingsDelete (otInstance *aInstance, uint16_t aKey, int aIndex)
```

Removes a setting from the setting store.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aKey	The key associated with the requested setting.
[in]	aIndex	The index of the value to be removed. If set to -1, all values for this `aKey` will be removed.

Deletes a specific value from the setting identified by aKey from the settings store.

Note that the underlying implementation is not required to maintain the order of the items associated with a specific key.

Definition at line `233` of file `include/openthread/platform/settings.h`

otPlatSettingsWipe

```
void otPlatSettingsWipe (otInstance *aInstance)
```

Removes all settings from the setting store.

Parameters

[in]	aInstance	The OpenThread instance structure.

Deletes all settings from the settings store, resetting it to its initial factory state.

Definition at line 243 of file include/openthread/platform/settings.h

SPI Slave

SPI Slave

This module includes the platform abstraction for SPI slave communication.

Typedefs

typedef bool(* otPlatSpiSlaveTransactionCompleteCallback)(void *aContext, uint8_t *aOutputBuf, uint16_t aOutputBufLen, uint8_t *aInputBuf, uint16_t aInputBufLen, uint16_t aTransactionLength)
Indicates that a SPI transaction has completed with the given length.

typedef void(* otPlatSpiSlaveTransactionProcessCallback)(void *aContext)
Invoked after a transaction complete callback is called and returns `TRUE` to do any further processing required.

Functions

otError otPlatSpiSlaveEnable(otPlatSpiSlaveTransactionCompleteCallback aCompleteCallback, otPlatSpiSlaveTransactionProcessCallback aProcessCallback, void *aContext)
Initialize the SPI slave interface.

void otPlatSpiSlaveDisable(void)
Shutdown and disable the SPI slave interface.

otError otPlatSpiSlavePrepareTransaction(uint8_t *aOutputBuf, uint16_t aOutputBufLen, uint8_t *aInputBuf, uint16_t aInputBufLen, bool aRequestTransactionFlag)
Prepare data for the next SPI transaction.

Typedef Documentation

otPlatSpiSlaveTransactionCompleteCallback

```
typedef bool(* otPlatSpiSlaveTransactionCompleteCallback) (void *aContext, uint8_t *aOutputBuf, uint16_t aOutputBufLen, uint8_t *aInputBuf, uint16_t aInputBufLen, uint16_t aTransactionLength) )(void *aContext, uint8_t *aOutputBuf, uint16_t aOutputBufLen, uint8_t *aInputBuf, uint16_t aInputBufLen, uint16_t aTransactionLength)
```

Indicates that a SPI transaction has completed with the given length.

Parameters

[in]	aContext	Context pointer passed into `otPlatSpiSlaveEnable()`.
[in]	aOutputBuf	Value of `aOutputBuf` from last call to `otPlatSpiSlavePrepareTransaction()`.
[in]	aOutputBufLen	Value of `aOutputBufLen` from last call to `otPlatSpiSlavePrepareTransaction()`.
[in]	aInputBuf	Value of aInputBuf from last call to `otPlatSpiSlavePrepareTransaction()`.
[in]	aInputBufLen	Value of aInputBufLen from last call to `otPlatSpiSlavePrepareTransaction()`
[in]	aTransactionLength	Length of the completed transaction, in bytes.

The data written to the slave has been written to the pointer indicated by the `aInputBuf` argument to the previous call to `otPlatSpiSlavePrepareTransaction()`.

Once this function is called, `otPlatSpiSlavePrepareTransaction()` is invalid and must be called again for the next transaction to be valid.

Note that this function is always called at the end of a transaction, even if `otPlatSpiSlavePrepareTransaction()` has not yet been called. In such cases, `aOutputBufLen` and `aInputBufLen` will be zero.

This callback can be called from ISR context. The return value from this function indicates if any further processing is required. If `TRUE` is returned the platform spi-slave driver implementation must invoke the transaction process callback (`aProcessCallback` set in `otPlatSpiSlaveEnable()`) which unlike this callback must be called from the same OS context that any other OpenThread API/callback is called.

Returns

- TRUE if after this call returns the platform should invoke the process callback `aProcessCallback`, FALSE if there is nothing to process and no need to invoke the process callback.

Definition at line `82` of file `include/openthread/platform/spi-slave.h`

otPlatSpiSlaveTransactionProcessCallback

```
typedef void(* otPlatSpiSlaveTransactionProcessCallback) (void *aContext) )(void *aContext)
```

Invoked after a transaction complete callback is called and returns `TRUE` to do any further processing required.

Parameters

[in]	aContext	Context pointer passed into `otPlatSpiSlaveEnable()`.

Unlike `otPlatSpiSlaveTransactionCompleteCallback` which can be called from any OS context (e.g., ISR), this callback MUST be called from the same OS context as any other OpenThread API/callback.

Definition at line `97` of file `include/openthread/platform/spi-slave.h`

Function Documentation

otPlatSpiSlaveEnable

```
otError otPlatSpiSlaveEnable (otPlatSpiSlaveTransactionCompleteCallback aCompleteCallback,
otPlatSpiSlaveTransactionProcessCallback aProcessCallback, void *aContext)
```

Initialize the SPI slave interface.

Parameters

[in]	aCompleteCallback	Pointer to transaction complete callback.
[in]	aProcessCallback	Pointer to process callback.
[in]	aContext	Context pointer to be passed to callbacks.

Note that SPI slave is not fully ready until a transaction is prepared using `otPlatSPISlavePrepareTransaction()`.

If `otPlatSPISlavePrepareTransaction() is not called before the master begins a transaction, the resulting SPI transaction will send all 0xFF` bytes and discard all received bytes.

Definition at line `116` of file `include/openthread/platform/spi-slave.h`

otPlatSpiSlaveDisable

> void otPlatSpiSlaveDisable (void)

Shutdown and disable the SPI slave interface.

Parameters

N/A		

Definition at line `123` of file `include/openthread/platform/spi-slave.h`

otPlatSpiSlavePrepareTransaction

> otError otPlatSpiSlavePrepareTransaction (uint8_t *aOutputBuf, uint16_t aOutputBufLen, uint8_t *aInputBuf, uint16_t aInputBufLen, bool aRequestTransactionFlag)

Prepare data for the next SPI transaction.

Parameters

[in]	aOutputBuf	Data to be written to MISO pin
[in]	aOutputBufLen	Size of the output buffer, in bytes
[in]	aInputBuf	Data to be read from MOSI pin
[in]	aInputBufLen	Size of the input buffer, in bytes
[in]	aRequestTransactionFlag	Set to true if host interrupt should be set

Data pointers MUST remain valid until the transaction complete callback is called by the SPI slave driver, or until after the next call to `otPlatSpiSlavePrepareTransaction()`.

May be called more than once before the SPI master initiates the transaction. Each **successful** call to this function will cause the previous values from earlier calls to be discarded.

Not calling this function after a completed transaction is the same as if this function was previously called with both buffer lengths set to zero and `aRequestTransactionFlag` set to `false`.

Once `aOutputBufLen` bytes of `aOutputBuf` has been clocked out, the MISO pin shall be set high until the master finishes the SPI transaction. This is the functional equivalent of padding the end of `aOutputBuf` with `0xFF` bytes out to the length of the transaction.

Once `aInputBufLen` bytes of aInputBuf have been clocked in from MOSI, all subsequent values from the MOSI pin are ignored until the SPI master finishes the transaction.

Note that even if `aInputBufLen` or `aOutputBufLen` (or both) are exhausted before the SPI master finishes a transaction, the ongoing size of the transaction must still be kept track of to be passed to the transaction complete callback. For example, if `aInputBufLen` is equal to 10 and `aOutputBufLen` equal to 20 and the SPI master clocks out 30 bytes, the value 30 is passed to the transaction complete callback.

If a `NULL` pointer is passed in as `aOutputBuf` or `aInputBuf` it means that that buffer pointer should not change from its previous/current value. In this case, the corresponding length argument should be ignored. For example, `otPlatSpiSlavePrepareTransaction(NULL, 0, aInputBuf, aInputLen, false)` changes the input buffer pointer and its length but keeps the output buffer pointer same as before.

Any call to this function while a transaction is in progress will cause all of the arguments to be ignored and the return value to be `OT_ERROR_BUSY`.

Definition at line `166` of file `include/openthread/platform/spi-slave.h`

Time Service

Time Service

This module includes the platform abstraction for the time service.

Functions

uint64_t	otPlatTimeGet(void) Get the current platform time in microseconds referenced to a continuous monotonic local clock (64 bits width).
uint16_t	otPlatTimeGetXtalAccuracy(void) Get the current estimated worst case accuracy (maximum ± deviation from the nominal frequency) of the local platform clock in units of PPM.

Function Documentation

otPlatTimeGet

```
uint64_t otPlatTimeGet (void)
```

Get the current platform time in microseconds referenced to a continuous monotonic local clock (64 bits width).

Parameters

N/A		

The clock SHALL NOT wrap during the device's uptime. Implementations SHALL therefore identify and compensate for internal counter overflows. The clock does not have a defined epoch and it SHALL NOT introduce any continuous or discontinuous adjustments (e.g. leap seconds). Implementations SHALL compensate for any sleep times of the device.

Implementations MAY choose to discipline the platform clock and compensate for sleep times by any means (e.g. by combining a high precision/low power RTC with a high resolution counter) as long as the exposed combined clock provides continuous monotonic microsecond resolution ticks within the accuracy limits announced by otPlatTimeGetXtalAccuracy.

Returns

- The current time in microseconds.

Definition at line `73` of file `include/openthread/platform/time.h`

otPlatTimeGetXtalAccuracy

```
uint16_t otPlatTimeGetXtalAccuracy (void)
```

Get the current estimated worst case accuracy (maximum ± deviation from the nominal frequency) of the local platform clock in units of PPM.

Parameters

N/A		

Note

Implementations MAY estimate this value based on current operating conditions (e.g. temperature).

In case the implementation does not estimate the current value but returns a fixed value, this value MUST be the worst-case accuracy over all possible foreseen operating conditions (temperature, pressure, etc) of the implementation.

Returns

- The current platform clock accuracy, in PPM.

Definition at line `90` of file `include/openthread/platform/time.h`

Toolchain

Toolchain

This module defines a toolchain abstraction layer through macros.

Usage:

```
typedef
OT_TOOL_PACKED_BEGIN
struct
{
   char mField1;
   union
   {
      char mField2;
      long mField3;
   } OT_TOOL_PACKED_FIELD;
} OT_TOOL_PACKED_END packed_struct_t;
```

Macros

#define OT_MUST_USE_RESULT undefined
Compiler-specific indication that a class or enum must be used when it is the return value of a function.

#define OT_TOOL_PACKED_BEGIN undefined
Compiler-specific indication that a class or struct must be byte packed.

#define OT_TOOL_PACKED_FIELD undefined
Indicate to the compiler a nested struct or union to be packed within byte packed class or struct.

#define OT_TOOL_WEAK undefined
Compiler-specific weak symbol modifier.

#define OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK (aFmtIndex, aStartIndex)
Specifies that a function or method takes `printf` style arguments and should be type-checked against a format string.

#define OT_UNUSED_VARIABLE (VARIABLE)
Suppress unused variable warning in specific toolchains.

#define OT_UNREACHABLE_CODE (CODE)
Suppress Unreachable code warning in specific toolchains.

#define OT_APPLE_IGNORE_GNU_FOLDING_CONSTANT (...)

#define OT_FALL_THROUGH undefined
Suppress fall through warning in specific compiler.

Macro Definition Documentation

OT_MUST_USE_RESULT

```
#define OT_MUST_USE_RESULT
```

Compiler-specific indication that a class or enum must be used when it is the return value of a function.

Note

- This is currently only available with clang (C++17 implements it as attribute [[nodiscard]]).
- To suppress the 'unused-result' warning/error, please use the '-Wno-unused-result' compiler option.

Definition at line 81 of file include/openthread/platform/toolchain.h

OT_TOOL_PACKED_BEGIN

```
#define OT_TOOL_PACKED_BEGIN
```

Compiler-specific indication that a class or struct must be byte packed.

Definition at line 176 of file include/openthread/platform/toolchain.h

OT_TOOL_PACKED_FIELD

```
#define OT_TOOL_PACKED_FIELD
```

Indicate to the compiler a nested struct or union to be packed within byte packed class or struct.

Definition at line 177 of file include/openthread/platform/toolchain.h

OT_TOOL_WEAK

```
#define OT_TOOL_WEAK
```

Compiler-specific weak symbol modifier.

Definition at line 179 of file include/openthread/platform/toolchain.h

OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK

```
#define OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK
```

Specifies that a function or method takes `printf` style arguments and should be type-checked against a format string.

Must be added after the function/method declaration. For example:

`void MyPrintf(void *aObject, const char *aFormat, ...) OT_TOOL_PRINTF_STYLE_FORMAT_ARG_CHECK(2, 3);`

The two argument index values indicate format string and first argument to check against it. They start at index 1 for the first parameter in a function and at index 2 for the first parameter in a method.

Definition at line 181 of file include/openthread/platform/toolchain.h

OT_UNUSED_VARIABLE

```
#define OT_UNUSED_VARIABLE
```

Value:

```
0    do                    \
0    {                     \
0       (void)(VARIABLE);        \
0    } while (false)
```

Suppress unused variable warning in specific toolchains.

Definition at line 252 of file include/openthread/platform/toolchain.h

OT_UNREACHABLE_CODE

```
#define OT_UNREACHABLE_CODE
```

Value:

```
(CODE)
```

Suppress Unreachable code warning in specific toolchains.

Definition at line 258 of file include/openthread/platform/toolchain.h

OT_APPLE_IGNORE_GNU_FOLDING_CONSTANT

```
#define OT_APPLE_IGNORE_GNU_FOLDING_CONSTANT
```

Value:

```
(...)
```

Definition at line 284 of file include/openthread/platform/toolchain.h

OT_FALL_THROUGH

```
#define OT_FALL_THROUGH
```

Value:

```
0    do           \
0    {            \
0    } while (false) /* fallthrough */
```

Suppress fall through warning in specific compiler.

Definition at line 300 of file include/openthread/platform/toolchain.h

TREL - Platform

TREL - Platform

This module includes the platform abstraction for Thread Radio Encapsulation Link (TREL) using DNS-SD and UDP/IPv6.

Modules

otPlatTrelPeerInfo

Typedefs

typedef struct otPlatTrelPeerInfo — otPlatTrelPeerInfo
Represents a TREL peer info discovered using DNS-SD browse on the service name "_trel._udp".

Functions

void otPlatTrelEnable(otInstance *aInstance, uint16_t *aUdpPort)
Initializes and enables TREL platform layer.

void otPlatTrelDisable(otInstance *aInstance)
Disables TREL platform layer.

void otPlatTrelHandleDiscoveredPeerInfo(otInstance *aInstance, const otPlatTrelPeerInfo *aInfo)
This is a callback function from platform layer to report a discovered TREL peer info.

void otPlatTrelRegisterService(otInstance *aInstance, uint16_t aPort, const uint8_t *aTxtData, uint8_t aTxtLength)
Registers a new service to be advertised using DNS-SD [RFC6763].

void otPlatTrelSend(otInstance *aInstance, const uint8_t *aUdpPayload, uint16_t aUdpPayloadLen, const otSockAddr *aDestSockAddr)
Requests a TREL UDP packet to be sent to a given destination.

void otPlatTrelHandleReceived(otInstance *aInstance, uint8_t *aBuffer, uint16_t aLength)
Is a callback from platform to notify of a received TREL UDP packet.

Typedef Documentation

otPlatTrelPeerInfo

```
typedef struct otPlatTrelPeerInfo otPlatTrelPeerInfo
```

Represents a TREL peer info discovered using DNS-SD browse on the service name "_trel._udp".

Definition at line 133 of file include/openthread/platform/trel.h

Function Documentation

otPlatTrelEnable

```
void otPlatTrelEnable (otInstance *aInstance, uint16_t *aUdpPort)
```

Initializes and enables TREL platform layer.

Parameters

[in]	aInstance	The OpenThread instance.
[out]	aUdpPort	A pointer to return the selected port number by platform layer.

Upon this call, the platform layer MUST perform the following:

1) TREL platform layer MUST open a UDP socket to listen for and receive TREL messages from peers. The socket is bound to an ephemeral port number chosen by the platform layer. The port number MUST be returned in `aUdpPort`. The socket is also bound to network interface(s) on which TREL is to be supported. The socket and the chosen port should stay valid while TREL is enabled.

2) Platform layer MUST initiate an ongoing DNS-SD browse on the service name "_trel._udp" within the local browsing domain to discover other devices supporting TREL. The ongoing browse will produce two different types of events: "add" events and "remove" events. When the browse is started, it should produce an "add" event for every TREL peer currently present on the network. Whenever a TREL peer goes offline, a "remove" event should be produced. "remove" events are not guaranteed, however. When a TREL service instance is discovered, a new ongoing DNS-SD query for an AAAA record should be started on the hostname indicated in the SRV record of the discovered instance. If multiple host IPv6 addressees are discovered for a peer, one with highest scope among all addresses MUST be reported (if there are multiple address at same scope, one must be selected randomly).

TREL platform MUST signal back the discovered peer info using `otPlatTrelHandleDiscoveredPeerInfo()` callback. This callback MUST be invoked when a new peer is discovered, when there is a change in an existing entry (e.g., new TXT record or new port number or new IPv6 address), or when the peer is removed.

Definition at line `87` of file `include/openthread/platform/trel.h`

otPlatTrelDisable

```
void otPlatTrelDisable (otInstance *aInstance)
```

Disables TREL platform layer.

Parameters

N/A	aInstance	

After this call, the platform layer MUST stop DNS-SD browse on the service name "_trel._udp", stop advertising the TREL DNS-SD service (from `otPlatTrelRegisterService()`) and MUST close the UDP socket used to receive TREL messages.

@pram[in] aInstance The OpenThread instance.

Definition at line `98` of file `include/openthread/platform/trel.h`

otPlatTrelHandleDiscoveredPeerInfo

```
void otPlatTrelHandleDiscoveredPeerInfo (otInstance *aInstance, const otPlatTrelPeerInfo *aInfo)
```

This is a callback function from platform layer to report a discovered TREL peer info.

Parameters

[in]	aInstance	The OpenThread instance.

[in]	aInfo	A pointer to the TREL peer info.

Note

- The `aInfo` structure and its content (e.g., the `mTxtData` buffer) does not need to persist after returning from this call. OpenThread code will make a copy of all the info it needs.

Definition at line `145` of file `include/openthread/platform/trel.h`

otPlatTrelRegisterService

```
void otPlatTrelRegisterService (otInstance *aInstance, uint16_t aPort, const uint8_t *aTxtData, uint8_t aTxtLength)
```

Registers a new service to be advertised using DNS-SD [RFC6763].

Parameters

[in]	aInstance	The OpenThread instance.
[in]	aPort	The port number to include in the SRV record of the advertised service.
[in]	aTxtData	A pointer to the TXT record data (encoded) to be include in the advertised service.
[in]	aTxtLength	The length of `aTxtData` (number of bytes).

The service name is "_trel._udp". The platform should use its own hostname, which when combined with the service name and the local DNS-SD domain name will produce the full service instance name, for example "example-host._trel._udp.local.".

The domain under which the service instance name appears will be 'local' for mDNS, and will be whatever domain is used for service registration in the case of a non-mDNS local DNS-SD service.

A subsequent call to this function updates the previous service. It is used to update the TXT record data and/or the port number.

The `aTxtData` buffer is not persisted after the return from this function. The platform layer MUST NOT keep the pointer and instead copy the content if needed.

Definition at line `170` of file `include/openthread/platform/trel.h`

otPlatTrelSend

```
void otPlatTrelSend (otInstance *aInstance, const uint8_t *aUdpPayload, uint16_t aUdpPayloadLen, const otSockAddr *aDestSockAddr)
```

Requests a TREL UDP packet to be sent to a given destination.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aUdpPayload	A pointer to UDP payload.
[in]	aUdpPayloadLen	The payload length (number of bytes).
[in]	aDestSockAddr	The destination socket address.

Definition at line `181` of file `include/openthread/platform/trel.h`

otPlatTrelHandleReceived

```
void otPlatTrelHandleReceived (otInstance *aInstance, uint8_t *aBuffer, uint16_t aLength)
```

Is a callback from platform to notify of a received TREL UDP packet.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aBuffer	A buffer containing the received UDP payload.
[in]	aLength	UDP payload length (number of bytes).

Note

- The buffer content (up to its specified length) may get changed during processing by OpenThread core (e.g., decrypted in place), so the platform implementation should expect that after returning from this function the `aBuffer` content may have been altered.

Definition at line `198` of file `include/openthread/platform/trel.h`

otPlatTrelPeerInfo

Represents a TREL peer info discovered using DNS-SD browse on the service name "_trel._udp".

Public Attributes

bool	mRemoved This boolean flag indicates whether the entry is being removed or added.
const uint8_t *	mTxtData The TXT record data (encoded as specified by DNS-SD) from the SRV record of the discovered TREL peer service instance.
uint16_t	mTxtLength Number of bytes in `mTxtData` buffer.
otSockAddr	mSockAddr The TREL peer socket address (IPv6 address and port number).

Public Attribute Documentation

mRemoved

```
bool otPlatTrelPeerInfo::mRemoved
```

This boolean flag indicates whether the entry is being removed or added.

- TRUE indicates that peer is removed.
- FALSE indicates that it is a new entry or an update to an existing entry.

Definition at line `113` of file `include/openthread/platform/trel.h`

mTxtData

```
const uint8_t* otPlatTrelPeerInfo::mTxtData
```

The TXT record data (encoded as specified by DNS-SD) from the SRV record of the discovered TREL peer service instance.

Definition at line `120` of file `include/openthread/platform/trel.h`

mTxtLength

```
uint16_t otPlatTrelPeerInfo::mTxtLength
```

Number of bytes in `mTxtData` buffer.

Definition at line `122` of file `include/openthread/platform/trel.h`

mSockAddr

> otSockAddr otPlatTrelPeerInfo::mSockAddr

The TREL peer socket address (IPv6 address and port number).

The port number is determined from the SRV record of the discovered TREL peer service instance. The IPv6 address is determined from the DNS-SD query for AAAA records on the hostname indicated in the SRV record of the discovered service instance. If multiple host IPv6 addressees are discovered, one with highest scope is used.

Definition at line `132` of file `include/openthread/platform/trel.h`

Infrastructure Interface

Infrastructure Interface

This module includes the platform abstraction for the adjacent infrastructure network interface.

Functions

bool otPlatInfraIfHasAddress(uint32_t aInfraIfIndex, const otIp6Address *aAddress)
Tells whether an infra interface has the given IPv6 address assigned.

otError otPlatInfraIfSendIcmp6Nd(uint32_t aInfraIfIndex, const otIp6Address *aDestAddress, const uint8_t *aBuffer, uint16_t aBufferLength)
Sends an ICMPv6 Neighbor Discovery message on given infrastructure interface.

void otPlatInfraIfRecvIcmp6Nd(otInstance *aInstance, uint32_t aInfraIfIndex, const otIp6Address *aSrcAddress, const uint8_t *aBuffer, uint16_t aBufferLength)
The infra interface driver calls this method to notify OpenThread that an ICMPv6 Neighbor Discovery message is received.

otError otPlatInfraIfStateChanged(otInstance *aInstance, uint32_t aInfraIfIndex, bool aIsRunning)
The infra interface driver calls this method to notify OpenThread of the interface state changes.

otError otPlatInfraIfDiscoverNat64Prefix(uint32_t aInfraIfIndex)
Send a request to discover the NAT64 prefix on the infrastructure interface with `aInfraIfIndex`.

void otPlatInfraIfDiscoverNat64PrefixDone(otInstance *aInstance, uint32_t aInfraIfIndex, const otIp6Prefix *aIp6Prefix)
The infra interface driver calls this method to notify OpenThread that the discovery of NAT64 prefix is done.

Function Documentation

otPlatInfraIfHasAddress

> bool otPlatInfraIfHasAddress (uint32_t aInfraIfIndex, const otIp6Address *aAddress)

Tells whether an infra interface has the given IPv6 address assigned.

Parameters

[in]	aInfraIfIndex	The index of the infra interface.
[in]	aAddress	The IPv6 address.

Returns

- TRUE if the infra interface has given IPv6 address assigned, FALSE otherwise.

Definition at line `68` of file `include/openthread/platform/infra_if.h`

otPlatInfraIfSendIcmp6Nd

> otError otPlatInfraIfSendIcmp6Nd (uint32_t aInfraIfIndex, const otIp6Address *aDestAddress, const uint8_t *aBuffer, uint16_t aBufferLength)

Sends an ICMPv6 Neighbor Discovery message on given infrastructure interface.

Parameters

[in]	aInfraIfIndex	The index of the infrastructure interface this message is sent to.
[in]	aDestAddress	The destination address this message is sent to.
[in]	aBuffer	The ICMPv6 message buffer. The ICMPv6 checksum is left zero and the platform should do the checksum calculate.
[in]	aBufferLength	The length of the message buffer.

See RFC 4861: https://tools.ietf.org/html/rfc4861.

Note

- Per RFC 4861, the implementation should send the message with IPv6 link-local source address of interface `aInfraIfIndex` and IP Hop Limit 255.

Definition at line `88` of file `include/openthread/platform/infra_if.h`

otPlatInfraIfRecvIcmp6Nd

```
void otPlatInfraIfRecvIcmp6Nd (otInstance *aInstance, uint32_t aInfraIfIndex, const otIp6Address *aSrcAddress, const uint8_t *aBuffer, uint16_t aBufferLength)
```

The infra interface driver calls this method to notify OpenThread that an ICMPv6 Neighbor Discovery message is received.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aInfraIfIndex	The index of the infrastructure interface on which the ICMPv6 message is received.
[in]	aSrcAddress	The source address this message is received from.
[in]	aBuffer	The ICMPv6 message buffer.
[in]	aBufferLength	The length of the ICMPv6 message buffer.

See RFC 4861: https://tools.ietf.org/html/rfc4861.

Note

- Per RFC 4861, the caller should enforce that the source address MUST be a IPv6 link-local address and the IP Hop Limit MUST be 255.

Definition at line `109` of file `include/openthread/platform/infra_if.h`

otPlatInfraIfStateChanged

```
otError otPlatInfraIfStateChanged (otInstance *aInstance, uint32_t aInfraIfIndex, bool aIsRunning)
```

The infra interface driver calls this method to notify OpenThread of the interface state changes.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aInfraIfIndex	The index of the infrastructure interface.
[in]	aIsRunning	A boolean that indicates whether the infrastructure interface is running.

It is fine for the platform to call to method even when the running state of the interface hasn't changed. In this case, the Routing Manager state is not affected.

Definition at line `134` of file `include/openthread/platform/infra_if.h`

otPlatInfraIfDiscoverNat64Prefix

```
otError otPlatInfraIfDiscoverNat64Prefix (uint32_t aInfraIfIndex)
```

Send a request to discover the NAT64 prefix on the infrastructure interface with `aInfraIfIndex`.

Parameters

[in]	aInfraIfIndex	The index of the infrastructure interface to discover the NAT64 prefix.

OpenThread will call this method periodically to monitor the presence or change of NAT64 prefix.

Definition at line `147` of file `include/openthread/platform/infra_if.h`

otPlatInfraIfDiscoverNat64PrefixDone

```
void otPlatInfraIfDiscoverNat64PrefixDone (otInstance *aInstance, uint32_t aInfraIfIndex, const otIp6Prefix *aIp6Prefix)
```

The infra interface driver calls this method to notify OpenThread that the discovery of NAT64 prefix is done.

Parameters

[in]	aInstance	The OpenThread instance structure.
[in]	aInfraIfIndex	The index of the infrastructure interface on which the NAT64 prefix is discovered.
[in]	aIp6Prefix	A pointer to NAT64 prefix.

Is expected to be invoked after calling otPlatInfraIfDiscoverNat64Prefix. If no NAT64 prefix is discovered, `aIp6Prefix` shall point to an empty prefix with zero length.

Definition at line `161` of file `include/openthread/platform/infra_if.h`