RTX

Interface Specification

Open source 6LoWPAN implementation for ULE FP and PP

Contents

1 Introduction 3
1.1 Scope 3
1.2 History 3
1.3 References 3
1.4 Terms & Abbreviations 3
2 System Overview 5
2.1 Scope of the of the 6LoWPAN Library 5
2.2 Limitations of the 6LoWPAN library 5
2.3 General Implementation Considerations 6
2.4 Test Interface 6
3 6LBR Interface 7
3.1 6LBR Specific API Functions 7
3.2 6LBR Specific Type Definitions 14
4 6LN Interface 16
4.1 6LN Specific API Functions 16
4.2 6LN Specific Type Definitions 25
5 Example on how the 6LoWPAN Library is interfaced 29

1 Introduction

1.1 Scope

The scope of this document is to define the interfaces for the 6LoWPAN library implementation for DECT ULE for both FP and PP.

1.2 History

Revision	Author	Issue Date	Comments
0.1	THK	1-JUL-15	Initial Revision
0.2	THK	10-Jul-15	Reviewed internally by RTX (JTP and CM)
1.0	THK	13-Jul-15	Released to ULE Alliance
1.1	THK	24-Jul-15	Reviewed by HSO
1.2	THK	10-Aug-15	Remarks from ULE Alliance
1.3	JJO	06-Jan-16	Changes made during development at RTX
1.4	SFC	01-Mar-16	Updated interfaces.

1.3 References

[1] **Title:** SoW - Open source 6LoWPAN implementation for ULE FP and PP
Author: Rasmus Fossa
Location: q:\Projects\UleAlliance\Simone\Specifications\ProductRequirements\StatementOfWork\SoW-6LoWPAN26may2015.pdf

[2] **Title:** Transmission of IPv6 Packets over DECT Ultra Low Energy draft-ietf-6lo-dect-ule
Author: P. Mariager et al.
Location: https://tools.ietf.org/wg/6lo/draft-ietf-6lo-dect-ule/

[3] **Title:** HAN-FUN library implementation
Author: Bithium S.A.
Location: https://github.com/ULE-Alliance/hanfun.git

[4] **Title:** Testing framework for a 6LoPWAN implementation over DECT ULE networks
Author: Offenburg University of Applied Sciences
Location: q:\Projects\UleAlliance\Simone\Specifications\ProductRequirements\StatementOfWork\FromOffenburg\SoW_ule_6lotest_v06.pdf

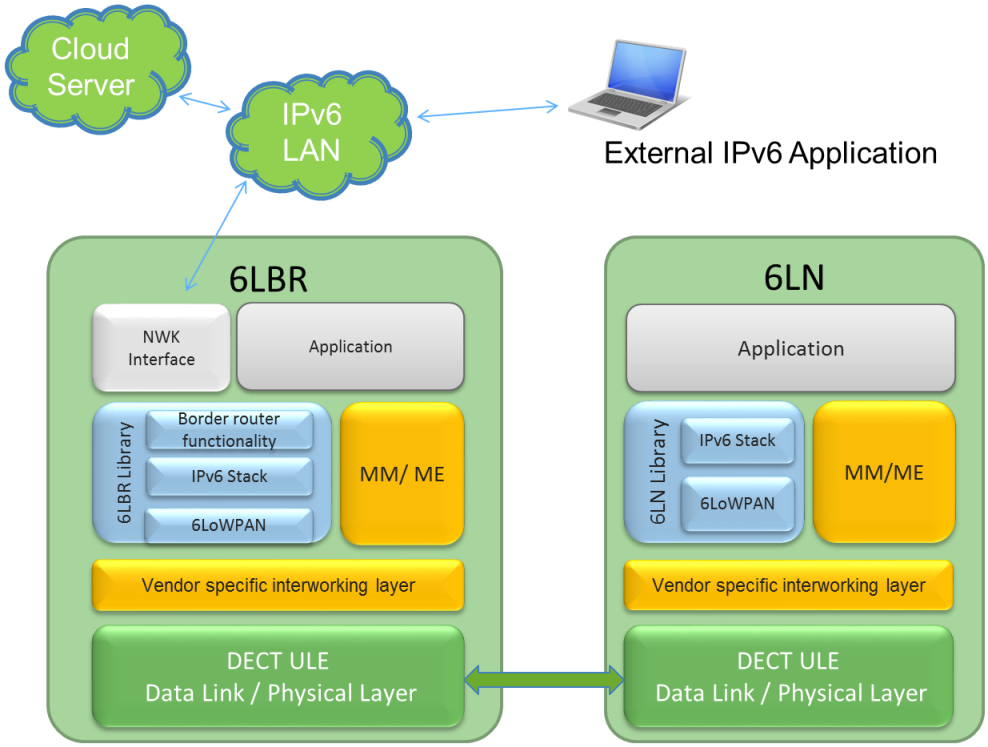
1.4 Terms & Abbreviations

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
6LBR	6LoWPAN Board Router (In DECT known as FP)

6LN	6LoWPAN Node (In DECT known as PP)
API	Application Interface
DECT	Digital Enhanced Cordless Telephone
FP	DECT Fixed Part (Base)
HS	DECT Handset
MMI	Man Machine Interface / User interface
PP	DECT Portable Part (Handset)
DNS	Domain Name System
ULE	DECT Ultra Low Energy
RTX	RTX A/S
SW	Software
HW	Hardware
FP	Fixed Part
PP	Portable Part
ME	Management Entity
MM	Mobility Management
HSO	Hochschule Offenburg (Offenburg University of Applied Sciences)
UA	ULE Alliance
LL	Link Layer, in this case the DECT ULE layer. (In some documents the DECT ULE layer is referred as Transport Layer, TL, which contradicts both the OSI and Internet model where the ULE layer would be Physical/Data Link layer and network layer respectively)

2 System Overview

The complete system consists of a DECT ULE Board Router (6LBR) with possible internet access and a number of DECT ULE nodes (6LN) using 6LowPAN as communication protocol, and DECT ULE as Phy/Data Link layer.



8

2.1 Scope of the of the 6LoWPAN Library

The library implements the procedures defined in [2]. The library can be configured either as for the node or for the gateway, respectively 6LN and 6LBR. It is based on the open source Contiki implementation of 6LoWPAN for 802.15.4, but with adaption for DECT-ULE as Link Layer.

The features implemented in the protocol stack are:
IPv6 header compression HC01. Supporting IPv6 only, UDP, TCP and ICMP. Supporting messages: RS, RA, NS, NA, context registration, SLAAC (based on IEEE MAC48), DNS resolver, addressing: global routable via registered context. Supported IP options: SLLAO, 6CO, ABRO, ARO, RDNSSO
Neighbor registration cache
Unicast / multicast (relay via 6LBR)

2.2 Limitations of the 6LoWPAN library

The 6LoWPAN library only handles the IPv6 configuration and packet transmission, thus all DECT ULE MM/Registration has to be handled elsewhere. Also power management such as sleep/wakeup handling is outside the scope of the 6LoWPAN library.

As specified in [2] the DECT ULE 6LoWPAN library will only support star topology network due to the nature of DECT ULE. Thus, multi-hop network is not supported and 6LN cannot be part of the routing.

Due to DECT ULE star topology, each branch of the star is considered an individual link and thus the 6LNs cannot directly hear one another and cannot talk to one another with link-local addresses, however communications directly between 6LN and 6LBR can use IPv6 link-local methodology.

2.3 General Implementation Considerations

The 6LoWPAN library is implemented in ANSI C, and will be provided as an open source library under the 3-clause BSD-style license. The 6LoWPAN library is based on the Contiki 6LowPAN implementation for 802.15.4, and will use the uIP stack and socket API from Contiki.

All the messages, parameters and data types defined in this document as in the 6LoWPAN implementation have little endian byte ordering and all the messages and data types with multiple members are byte aligned (packed).

All functions defined in this document are implemented as non-blocking. Pointers parsed as argument or given as return value must be considered as volatile, and responsibility of the caller to copy data.

2.3.1 Random generation of iid

Since the iid is based on random generation it is important that the random init function is seeded with a truly random number. random_init() must be called after **ule6loGI_init(..)** at the LBR and after **lla_init(..)** at the node.

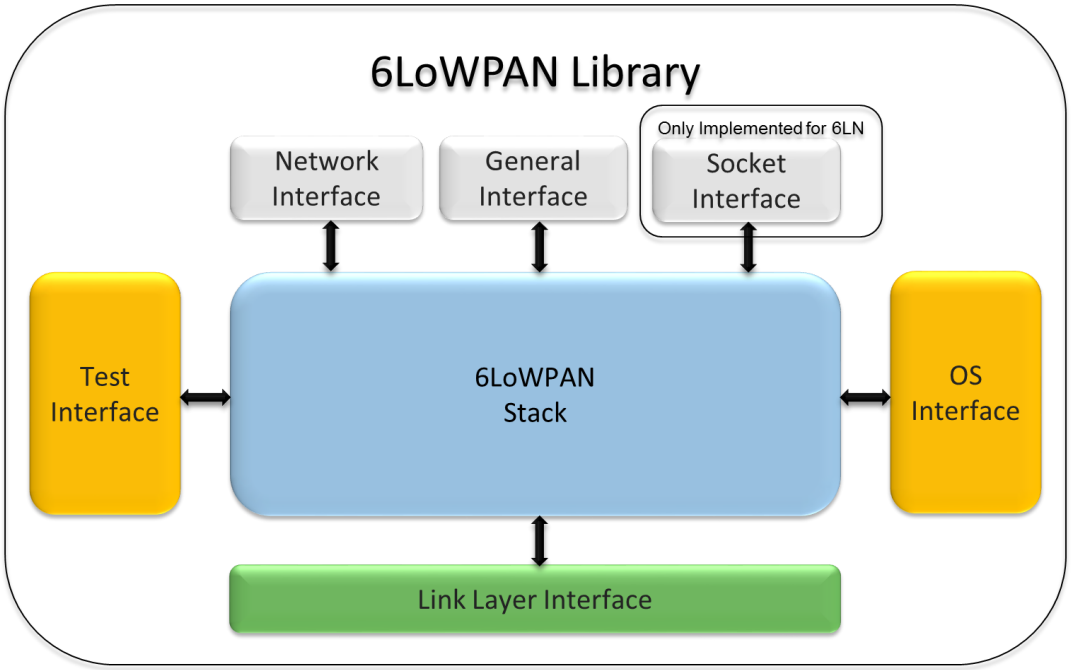
2.4 Test Interface

In order to provide a reliable software library testing of functionalities and interfaces are crucial. Therefore a test interface is implemented providing the necessary functions for both doing *White-box* and *black-box* testing. The test interface conforms to the requirement setup in [4], thereby enabling HSO to perform system test and final Acceptance testing.

3 6LBR Interface

3.1 6LBR Specific API Functions

The 6LBR interfaces upwards to an external network and a possible application and downwards to the DECT-ULE layer. Furthermore the 6LBR 6LoWPAN library also provides a test interface, which interacts directly into the library. Potentially the 6LoWPAN library could also provide a socket interface to the application. However this not included on the 6LBR side in the current version, for more option see, Socket Interface for the 6LN.



3.1.1 General Interface

The 6LoWPAN library provides a set of general interface functions for initializing the library and to get various parameters.

3.1.1.1 ule6loGI_init

Description: This function is called by the application/OS to initialize the 6LoWPAN library

Returns: `ule6lo_status_t`

Parameters:

Type	Name	Description
`const ule6lo_IPEI_t *`	`ULEAddr`	Pointer to IPEI of ULE device

3.1.1.2 ule6loGI_getStatus

Description: Returns status of 6LoWPAN library, `STATUS_SUCCESS` for working, otherwise not working

Returns: `ule6lo_status_t`

Parameters: None

3.1.1.3 ule6loGI_getIp6addr

Description: Function for getting IP address
Returns: `ule6lo_status_t`
Parameters:

Type	Name	Description
`ule6lo_ipType_t`	`ipType`	Type of IP address
`ule6lo_ip6addr_t*`	`ipAddr`	Pointer to IP address
`ule6lo_ipMode_t`	`mode`	Mode of IP address requested

3.1.1.4 ule6loGI_addContext

Description: Function called from the application to add a context address for compression.
Returns: `ule6lo_status_t`
Parameters:

Type	Name	Description
`uint8_t[]`	`prefix`	Array of bytes for the prefix
`uint8_t`	`prefixlength`	Length of prefix in bytes

3.1.1.5 ule6loGI_addMulticastAddr

Description: Function called from the application to add a multicast address to listen at.
Returns: `ule6lo_status_t`
Parameters:

Type	Name	Description
`ule6lo_ip6addr_t*`	`ipaddress`	Pointer to multicast address

3.1.1.6 ule6loGI_removeMulticastAddr

Description: Function called from the application to remove a multicast address we are currently listen at.
Returns: `ule6lo_status_t`
Parameters:

Type	Name	Description
`ule6lo_ip6addr_t*`	`ipaddress`	Pointer to multicast address

3.1.1.7 ule6loGI_setMacAddress

Description: Function to add the mac address for the LBR
Returns: `none`
Parameters:

Type	Name	Description
`ule6lo_macAddr_t*`	`macaddress`	Pointer to the mac address

~~3.1.1.8 ule6loGI_getDomain~~

~~Description: Returns a pointer to zero-terminated string with the domain name~~
~~Returns: `ule6lo_status_t`~~
~~Parameters:~~

~~Type~~	~~Name~~	~~Description~~
~~`ule6lo_ip6addr_t*`~~	~~`ipAddr`~~	~~Pointer to IP address~~
~~`char *`~~	~~`domain`~~	~~Pointer to zero-terminated string~~

3.1.2 Network Interface

The 6LoWPAN library on the 6LBR interfaces upwards to the network. This interface provide the following functions:

3.1.2.1 ule6loNI_receive

Description: This function is called by the network interface to deliver an incoming IPv6 packet to the 6LoWPAN library. The packet is on layer 2, including mac header.

Returns: `ule6lo_status_t`

Parameters:

Type	Name	Description
`const uint8_t *`	`Data`	Pointer to IPv6 packet
`uint16_t`	`dataLength`	Length of IPv6 packet

3.1.2.2 ule6loNI_send

Description: This function is called by 6LoWPAN library to deliver an outgoing IPv6 packet to the network. The packet is on layer 2, including mac header.

Returns: `ule6lo_status_t`

Parameters:

Type	Name	Description
`const uint8_t *`	`data`	Pointer to IPv6 packet
`uint16_t`	`dataLength`	Length of IPv6 packet

3.1.2.3 ule6loNI_echoRequest

Description: This function sends out an echo request(ICMP 128) to the network.

Returns: None

Parameters: None

3.1.3 ULE Link Layer Interface

The 6LoWPAN library aligns the interface towards the LL with the interface used in the Han-Fun library [3], in order to obtain best possible interoperability. The Han-Fun defines following functions, where similar functions will be used for the 6LoWPAN library:

- Transport::initialize()
- Transport::receive()
- Transport::connected()
- Transport::delivered()
- Transport::send()

3.1.3.1 ule6loLLI_init

Description: Called after location registration to inform the 6LoWPAN library a new ULE link is connected

Returns: Status

Parameters:

Type	Name	Description
`const ule6lo_IPEI_t *`	`ULEAddr`	Pointer to IPEI of ULE device

3.1.3.2 ule6loLLI_receive

Description: Called from LL when LL receives a data indication and forwards data to the 6LoWPAN library.

Returns: status_t

Parameters:

Type	Name	Description
`const uint8_t *`	`data`	Pointer to ULE packet
`uint16_t`	`dataLength`	Length of ULE packet
`const ule6lo_IPEI_t *`	`ULEAddr`	Pointer to IPEI of ULE device

3.1.3.3 ule6loLLI_send

Description: Called from within 6LoWPAN library when sending a ULE packages to the LL layer

Returns: None

Parameters:

Type	Name	Description
`const uint8_t *`	`data`	Pointer to ULE packet
`uint16_t`	`dataLength`	Length of ULE packet
`const ule6lo_IPEI_t *`	`ULEAddr`	Pointer to IPEI of ULE device

3.1.3.4 ule6loLLI_delivered

Description: Called from LL after transmission to indicate status.
Returns: None
Parameters:

Type	Name	Description
`ule6lo_status_t`	`status`	Status of previous transmission
`const ule6lo_IPEI_t *`	`ULEAddr`	Pointer to IPEI of ULE device

3.1.4 OS Interface

3.1.4.1 ule6loOS_processRun

Description: This function should be called repeatedly from the main() program / OS to actually run the 6LoWPAN library.
Returns: None
Parameters: None

3.1.4.2 ule6loOS_getMACAddr

Description: Returns the 6LBRs MAC address. This function is hardware/ OS depended and needs to be implemented correspondingly
Returns: `ule6lo_macAddr_t *`
Parameters: None

3.1.4.3 ule6loOS_getTimerTick

Description: Function called by the 6LoWPAN library to get the system timer tick. The function expects a tick increment corresponding to10 ms. The function is hardware/ OS depended and needs to be implemented correspondingly.
Returns: `uint32_t`
Parameters: None

3.1.5 Test Interface

The following defines the test interface for the 6LBR 6LoWPAN library.

3.1.5.1 ule6loTestIn_init

Description: Initialize test interface (allocates buffers, resets packet counts and status) and returns status
Returns: `ule6lo_status_t`
Parameters: None

3.1.5.2 ule6loTestIn_deinit

Description: Terminates test interface, including cleanup of buffer handling and deregister call backs, returns status
Returns: `ule6lo_status_t`
Parameters: None

3.1.5.3 ule6loTestIn_reset

Description: Performs soft reset which emulates a hardware reset, everything is cleared including IPs and NB lists. Returns status
Returns: `ule6lo_status_t`
Parameters: None

3.1.5.4 ule6loTestIn_getNbListSize

Description: Returns the length of Neighbor list.
Returns: `uint16_t`
Parameters: None

3.1.5.5 ule6loTestIn_getNbList

Description: Copies specified index of the Neighbor list to specified destination
Returns: `ule6lo_status_t`
Parameters:

Type	Name	Description
`uint16_t`	`index`	Index in the neighbor list to be copied
`ule6lo_nbr_t*`	`nBListItem`	Item in neighbor list

3.1.5.6 ule6loTestIn_getnofSentPacket

Description: Returns amount of sent packets on DECT interface
Returns: `uint32_t`
Parameters: None

3.1.5.7 ule6loTestIn_getnofReceivedPacket

Description: Returns amount of received packets on DECT interface
Returns: `uint32_t`
Parameters: None

3.1.5.8 ule6loTestIn_regRxHook

Description: Register a hook function to be called every time an packet is received on the wireless interface, with the raw packet as argument

Returns: None

Parameters:

Type	Name	Description
`ule6lo_hock_t`	`rxHook`	Pointer callback function. This function is called every time a packet is received, with a pointer to the packet and length of the packet

3.1.5.9 ule6loTestIn_regTxHook

Description: Register a hook function to be called every time an packet is transmitted on the wireless interface, with the raw packet as argument

Returns: None

Parameters:

Type	Name	Description
`ule6lo_hock_t`	`txHook`	Pointer callback function. This function is called every time a packet is received, with a pointer to the packet and length of the packet

3.1.5.10 ule6loTestIn_regBorderRouterRxHook

Description: Register a hook function to be called every time an packet is received on the border router interface, with the raw packet as argument

Returns: None

Parameters:

Type	Name	Description
`ule6lo_hock_t`	`rxHook`	Pointer callback function. This function is called every time a packet is received, with a pointer to the packet and length of the packet

3.1.5.11 ule6loTestIn_regBorderRouterTxHook

Description: Register a hook function to be called every time an packet is transmitted on the border router interface, with the raw packet as argument

Returns: None

Parameters:

Type	Name	Description
`ule6lo_hock_t`	`txHook`	Pointer callback function. This function is called every time a packet is received, with a pointer to the packet and length of the packet

3.1.5.12 ule6loTestIn_getnofBorderRouterSentPacket

Description: Returns amount of sent packets on border router interface
Returns: `uint32_t`
Parameters: None

3.1.5.13 ule6loTestIn_getnofBorderRouterReceivedPacket

Description: Returns amount of received packets on border router interface
Returns: `uint32_t`
Parameters: None

3.2 6LBR Specific Type Definitions

3.2.1 ule6lo_status_t

Description: General status type
C-syntax:

```
typedef enum ule6lo_status_t
{
     STATUS_SUCCESS                = 0x00,     The request completed successfully.
     STATUS_NOT_CONNENCTED         = 0x01,     Connected
     STATUS_ERROR                  = 0x02,     Error
     STATUS_NO_DEVICE              = 0x03,     No such device
     STATUS_NO_DATA                = 0x04,     No data is available.
     STATUS_NOT_READY              = 0x05,     The device is not ready.
     STATUS_MAX                    = 0xFF
} ule6lo_ststaus_t;
```

3.2.2 ule6lo_ipType_t

Description: IP address type. More migth be added
C-syntax:

```
typedef enum ule6lo_ipType_t
{
     IP_ADDRESS_LINK_LOCAL         = 0x00,     Link local IP address
     IP_ADDRESS_GLOBAL             = 0x01,     Global IP address
     IP_ADDRESS_MAX                = 0xFF
} ule6lo_ipType_t;
```

3.2.3 ule6lo_macAddr_t

Description: MAC address type
C-syntax:

```
typedef union {
{
   uint8_t           u8[6];
} ule6lo_macAddr_t;
```

3.2.4 ule6lo_ip6addr_t

Description: IPv6 address type

C-syntax:

```
typedef union ule6lo_ip6addr_t {
{
   uint8_t          u8[16];
   uint16_t         u16[8];
} ule6lo_ip6addr_t;
```

3.2.5 ule6lo_IPEI_t

Description: IPEI address type

C-syntax:

```
typedef union {
{
   uint8_t          id[5];
} ule6lo_IPEI_t;
```

3.2.6 ule6lo_nbr_t

Description: An entry in the neighbor cache

C-syntax:

```
typedef struct ule6lo_ds6_nbr {
{
   ule6lo_ipaddr_t   ipaddr;
   ule6lo_IPEI_t     lladdr;
} ule6lo_nbr_t;
```

3.2.7 ule6lo_hook_t

Description: Hock function pointer type

C-syntax:

```
typedef void (*ule6lo_hook_t)(uint8_t *data, uint16_t dataLength)
```

3.2.8 ule6lo_ipMode_t

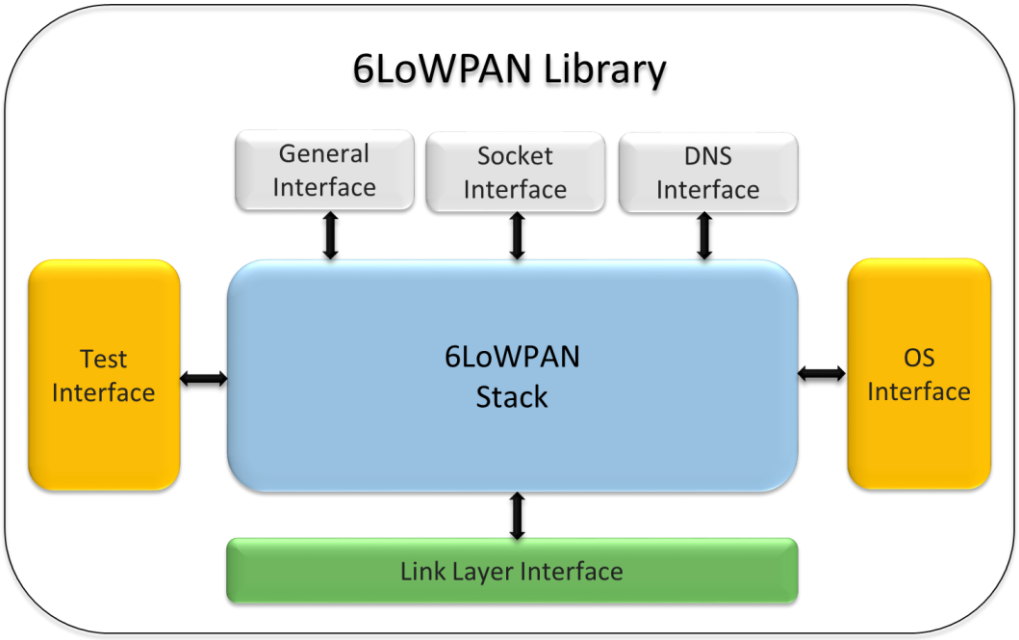
Description: IP address mode.

C-syntax:

```
typedef enum ule6lo_ipMode_t
{
     IP_ADDRESS_ANY               = -1,       ANY
     IP_ADDRESS_TENTATIVE         = 0,        Tentative
     IP_ADDRESS_PREFERRED         = 1,        Preferred
     IP_ADDRESS_DEPRECATED        = 2         Deprecated
} ule6lo_ipMode_t;
```

4 6LN Interface

The 6LoWPAN library on the ULE node interface upwards to the application with a BSD like socket interface, a DNS interface and a more general application interface and downwards to the DECT ULE layer. Also for the node a test interface is defined together with an OS abstraction interface.



4.1 6LN Specific API Functions

4.1.1 General Interface

The 6LoWPAN library provides a set of general interface functions for initializing the library and for getting various parameters.

4.1.1.1 ule6loGI_init

Description: This function is called by the application/OS to initialize the 6LoWPAN library
Returns: `ule6lo_status_t`
Parameters: None

4.1.1.2 ule6loGI_getStatus

Description: Returns status of 6LoWPAN library, `STATUS_SUCCESS` for working, otherwise not working
Returns: `ule6lo_status_t`
Parameters: None

4.1.1.3 ule6lo_getIp6addr

Description: Function for getting IP address
Returns: `ule6lo_status_t`
Parameters:

Type	Name	Description
`ule6lo_ipType_t`	`ipType`	Type of IP address
`ule6lo_ip6addr_t*`	`ipAddr`	Pointer to IP address
`ule6lo_ipMode_t`	`mode`	Mode of IP address requested

4.1.1.4 ule6loGI_addMulticastAddr

Description: Function to add a multicast address to listen at.
Returns: `ule6lo_status_t`
Parameters:

Type	Name	Description
`ule6lo_ip6addr_t*`	`ipaddress`	Pointer to multicast address

4.1.1.5 ule6loGI_removeMulticastAddr

Description: Function to remove a multicast address we are currently listen at.
Returns: `ule6lo_status_t`
Parameters:

Type	Name	Description
`ule6lo_ip6addr_t*`	`ipaddress`	Pointer to multicast address

4.1.1.6 ule6loGI_setMacAddress

Description: Function to add the mac address for the node
Returns: `none`
Parameters:

Type	Name	Description
`ule6lo_macAddr_t*`	`macaddress`	Pointer to the mac address

~~4.1.1.7 ule6loGI_getDomain~~

~~Description: Returns a pointer to zero-terminated string with the domain name~~
~~Returns: `ule6lo_status_t`~~
~~Parameters:~~

~~Type~~	~~Name~~	~~Description~~
~~`ule6lo_ip6addr_t*`~~	~~`ipAddr`~~	~~Pointer to IP address~~
~~`char *`~~	~~`domain`~~	~~Pointer to zero terminated string~~

4.1.2 Socket Interface

The socket interface upwards the application uses the BSD like implementation from Contiki, which provides both a TCP and UDP socket API. Therefore types and function definitions corresponds to the ones used in Contiki.

4.1.2.1 tcp_socket_register

Description: This function registers a TCP socket. The function sets up the output and input buffers for the socket and callback pointers

Returns: `Int`

Parameters:

Type	Name	Description
`struct tcp_socket *`	`S`	
`void *`	`Ptr`	
`uint8_t`	`input_databuf`	
`int`	`input_databuf_len`	
`uint8_t`	`output_databuf`	
`int`	`output_databuf_len`	
`tcp_socket_data_callback_t`	`input_callback`	
`tcp_socket_data_callback_t`	`event_callback`	

4.1.2.2 tcp_socket_connect

Description: Connects a TCP socket to an IP address and port number, returns -1 for failure.

Returns: `int`

Parameters:

Type	Name	Description
`struct tcp_socket*`	`S`	Pointer to TCP socket
`const uip_ipaddr_t*`	`Ipaddr`	Destination IP
`uint16_t port`	`port`	Destination port

4.1.2.3 tcp_socket_listen

Description: Listen to a TCP socket on specified port number, returns -1 for failure.

Returns: `int`

Parameters:

Type	Name	Description
`struct tcp_socket*`	`S`	Pointer to TCP socket
`uint16_t port`	`port`	Port number

4.1.2.4 tcp_socket_unlisten

Description: Stops listing to specified TCP socket, returns -1 for failure.

Returns: `int`

Parameters:

Type	Name	Description
`struct tcp_socket*`	`S`	Pointer to TCP socket

4.1.2.5 tcp_socket_send

Description: Sends data on specified TCP socket, returns -1 for failure.
Returns: `int`
Parameters:

Type	Name	Description
`struct tcp_socket*`	`S`	Pointer to TCP socket
`const uint8_t *`	`data`	Pointer to data to send
`int`	`datalen`	Length of data

4.1.2.6 tcp_socket_send_str

Description: Sends a string on specified TCP socket, string needs to be NULL terminated, returns -1 for failure.
Returns: `int`
Parameters:

Type	Name	Description
`struct tcp_socket*`	`S`	Pointer to TCP socket
`const char *`	`Str`	string to send

4.1.2.7 tcp_socket_close

Description: Closes connection on specified TCP socket.
Returns: `int`
Parameters:

Type	Name	Description
`struct tcp_socket*`	`S`	Pointer to TCP socket

4.1.2.8 tcp_socket_unregister

Description: Clean up TCP socket.
Returns: `int`
Parameters:

Type	Name	Description
`struct tcp_socket*`	`S`	Pointer to TCP socket

4.1.2.9 udp_socket_register

Description: This function registers the UDP socket with the system. A UDP socket must be registered before any data can be sent or received over the socket.
Returns: `int`
Parameters:

Type	Name	Description
`struct udp_socket *`	`S`	
`void *`	`Ptr`	
`udp_socket_data_callback_t`	`input_callback`	

4.1.2.10 udp_socket_close

Description: Closes and removes UDP socket
Returns: `int`
Parameters:

Type	Name	Description
`struct udp_socket*`	`c`	Pointer to UDP socket

4.1.2.11 udp_socket_bind

Description: Binds UDP socket to specified port number
Returns: `int`
Parameters:

Type	Name	Description
`struct udp_socket*`	`c`	Pointer to UDP socket
`uint16_t`	`Local_port`	Port number

4.1.2.12 udp_socket_connect

Description: Connects UDP socket (this is optional in UDP)
Returns: `int`
Parameters:

Type	Name	Description
`struct udp_socket*`	`c`	Pointer to UDP socket
`uip_ipaddr_t *`	`remote_addr`	Pointer to remote IP address
`uint16_t`	`Local_port`	Port number

4.1.2.13 udp_socket_send

Description: Sends data packet on specified UDP socket, needs to be in a "connected state" in order to know the recipient
Returns: `int`
Parameters:

Type	Name	Description
`struct udp_socket*`	`c`	Pointer to UDP socket
`const void *`	`data`	Pointer to data to send
`uint16_t`	`datalen`	Length of data

4.1.2.14 udp_socket_sendto

Description: Sends data packet on specified UDP socket, to specified IP address and port number

Returns: `int`

Parameters:

Type	Name	Description
`struct udp_socket*`	`c`	Pointer to UDP socket
`const void *`	`data`	Pointer to data to send
`uint16_t`	`datalen`	Length of data
`uip_ipaddr_t *`	`remote_addr`	Pointer to remote IP address
`uint16_t`	`Local_port`	Port number

4.1.3 DNS Interface

The 6LoWPAN library provides a set of DNS resolver functions used to lookup a hostname and map it to a numerical IP address. These functions are directly based on Contiki, thus types and function definitions corresponds to the ones used in Contiki.

4.1.3.1 resolv_query

Description: Queues a name so that a question for the name will be sent out.

Returns: None

Parameters:

Type	Name	Description
`const char *`	`name`	The hostname that is to be queried

4.1.3.2 resolv_lookup

Description: Look up a hostname in the array of known hostnames.

Returns: `resolv_status_t`

Parameters:

Type	Name	Description
`const char *`	`name`	The hostname that is to be queried
`uip_ipaddr_t **`	`ipaddr`	IP address corresponding to the hostname

4.1.4 ULE Link Layer Interface

The 6LoWPAN library aligns the interface towards the LL with the interface used in the Han-Fun library [3], in order to obtain best possible interoperability. The Han-Fun defines following functions, where similar function will be used for the 6LoWPAN library:

- Transport::initialize()
- Transport::receive()
- Transport::connected()

- Transport::check()
- Transport::delivered()
- Transport::send()

4.1.4.1 ule6loLLI_init

Description: Called after location registration to inform the 6LoWPAN library that the 6LN is now connected to 6LBR

Returns: `ule6lo_status_t`

Parameters:

Type	Name	Description
`const ule6lo_IPEI_t *`	`ULEAddr`	Pointer to IPEI of ULE device

4.1.4.2 ule6loLLI_receive

Description: Called from LL when LL receives a data indication and forwards data to the 6LoWPAN library.

Returns: `ule6lo_status_t`

Parameters:

Type	Name	Description
`const uint8_t *`	`data`	Pointer to ULE packet
`uint16_t`	`dataLength`	Length of ULE packet
`Const ule6lo_IPEI_t`	`*ULEAddr`	Ipei of the sending LBR

4.1.4.3 ule6loLLI_connected

Description: Called from LL to indicate status of connection, if ready to receive/transmit data, or disconnected/offline

Returns: None

Parameters:

Type	Name	Description
`ule6lo_status_t`	`status`	Status of the connection

4.1.4.4 ule6loLLI_check

Description: Called from LL before receiving data. It sends an empty string.

Returns: None

Parameters: None

4.1.4.5 ule6loLLI_send

Description: Called from the 6LoWPAN library when sending a ULE packages to the LL layer

Returns: None

Parameters:

Type	Name	Description
`const uint8_t *`	`data`	Pointer to ULE packet
`uint16_t`	`dataLength`	Length of ULE packet

4.1.4.6 ule6loLLI_delivered

Description: Called from LL after transmission to indicate status.
Returns: None
Parameters:

Type	Name	Description
`ule6lo_status_t`	`status`	Status of previous transmission

4.1.5 OS Interface

4.1.5.1 ule6loOS_processRun

Description: This function should be called repeatedly from the main() program / OS to actually run the 6LoWPAN library.
Returns: None
Parameters: None

4.1.5.2 ule6loOS_getMACAddr

Description: Returns the 6LNs MAC address. This function is hardware/ OS depended and needs to be implemented correspondingly
Returns: `ule6lo_macaddr_t *`
Parameters: None

4.1.5.3 ule6loOS_getTimerTick

Description: Function called by the 6LoWPAN library to get the system timer tick. The function expects a tick increment corresponding to10 ms. The function is hardware/ OS depended and needs to be implemented correspondingly.
Returns: `uint32_t`
Parameters: None

4.1.6 Test Interface

The following defines the test interface for the 6LN 6LoWPAN library:

4.1.6.1 ule6loTestIn_init

Description: Initialize test interface, returns status
Returns: `ule6lo_status_t`
Parameters: None

4.1.6.2 ule6loTestIn_deinit

Description: Terminates test interface, including cleanup of buffer handling and deregister call backs, returns status
Returns: `ule6lo_status_t`
Parameters: None

4.1.6.3 ule6loTestIn_reset

Description: Performs soft reset which emulates a hardware reset, everything is cleared including IPs and NB lists. Returns status
Returns: `ule6lo_status_t`
Parameters: None

4.1.6.4 ule6loTestIn_getNbListSize

Description: Returns the length of Neighbor list.
Returns: `uint16_t`
Parameters: None

4.1.6.5 ule6loTestIn_getNbList

Description: Copies specified index of the Neighbor list to specified destination
Returns: `ule6lo_status_t`
Parameters:

Type	Name	Description
`uint16_t`	`index`	Index of the neighbor list to be copied
`ule6lo_nbr_t*`	`nBListItem`	Item in neighbor list

4.1.6.6 ule6loTestIn_getnofSentPacket

Description: Returns amount of sent packets
Returns: `uint32_t`
Parameters: None

4.1.6.7 ule6loTestIn_getnofReceivedPacket

Description: Returns amount of received packets
Returns: `uint32_t`
Parameters: None

4.1.6.8 ule6loTestIn_regRxHook

Description: Register a hook function to be called, every time an packet is received, with the raw packet and length as argument

Returns: None

Parameters:

Type	Name	Description
`ule6lo_hock_t`	`rxHook`	Pointer callback function. This function is called every time a packet is received, with a pointer to the packet and length of the packet

4.1.6.9 ule6loTestIn_regTxHook

Description: Register a hook function to be called every time an packet is transmitted, with the raw packet as argument

Returns: None

Parameters:

Type	Name	Description
`ule6lo_hock_t`	`txHook`	Pointer callback function. This function is called every time a packet is received, with a pointer to the packet and length of the packet

4.2 6LN Specific Type Definitions

4.2.1 ule6lo_status_t

Description: General status type

C-syntax:

```
typedef enum ule6lo_status_t
{
     STATUS_SUCCESS                = 0x00,     The request completed successfully.
     STATUS_NOT_CONNENCTED         = 0x01,     Connected
     STATUS_ERROR                  = 0x02,     Error
     STATUS_NO_DEVICE              = 0x03,     No such device
     STATUS_NO_DATA                = 0x04,     No data is available.
     STATUS_NOT_READY              = 0x05,     The device is not ready.
     STATUS_MAX                    = 0xFF
} ule6lo_ststaus_t;
```

4.2.2 ule6lo_ipType_t

Description: IP address type. More migth be added

C-syntax:

```
typedef enum ule6lo_ipType_t
{
     IP_ADDRESS_LINK_LOCAL         = 0x00,     Link local IP address
     IP_ADDRESS_GLOBAL             = 0x01,     Global IP address
     IP_ADDRESS_MAX                = 0xFF
} ule6lo_ipType_t;
```

4.2.3 ule6lo_macAddr_t

Description: MAC address type
C-syntax:

```
typedef union {
{
   uint8_t          u8[6];
} ule6lo_macAddr_t;
```

4.2.4 ule6lo_ip6addr_t

Description: IPv6 address type
C-syntax:

```
typedef union ule6lo_ip6addr_t {
{
   uint8_t          u8[16];
   uint16_t         u16[8];
} ule6lo_ip6addr_t;
```

4.2.5 ule6lo_IPEI_t

Description: IPEI address type
C-syntax:

```
typedef union {
{
   uint8_t          id[5];
} ule6lo_IPEI_t;
```

4.2.6 ule6lo_hook_t

Description: Hook function pointer type
C-syntax:

```
typedef void (*ule6lo_hook_t)(uint8_t *data, uint16_t dataLength)
```

4.2.7 tcp_socket_event_callback_t

Description: TCP event callback function

C-syntax:

```
typedef enum {
  TCP_SOCKET_CONNECTED,
  TCP_SOCKET_CLOSED,
  TCP_SOCKET_TIMEDOUT,
  TCP_SOCKET_ABORTED,
  TCP_SOCKET_DATA_SENT
} tcp_socket_event_t;

Typedef void (* tcp_socket_event_callback_t)
(
   struct tcp_socket *          S,
   void *                       Ptr,
   tcp_socket_event_t           Event,
);
```

4.2.8 udp_socket

Description: UDP socket type

C-syntax:

```
struct udp_socket
{
   udp_socket_input_callback_t  input_callback;
   void *                       ptr
   struct process *             p
   Struct uip_conn *            C
};
```

4.2.9 udp_socket_input_callback_t

Description: A UDP socket callback function

C-syntax:

```
typedef void (* udp_socket_input_callback_t)
[
   struct udp_socket *          C,
   void *                       Ptr,
   const uip_ipaddr_t *         source_addr,
   uint16_t                     source_port,
   const uip_ipaddr_t *         dest_addr,
   uint16_t                     dest_port,
   const uint8_t *              input_data_ptr,
   int                          input_data_len,
);
```

4.2.10 resolv_status_t

Description: resolv status type

C-syntax:

```
typedef uint8_t resolv_status_t;
enum
{
   RESOLV_STATUS_CACHED            = 0x00,   Hostname is fresh and usable. This response is cached and will eventually
                                             expire to RESOLV_STATUS_EXPIRED.
   RESOLV_STATUS_UNCACHED          ,         Hostname was not found in the cache. Use resolv_query() to look it up.
   RESOLV_STATUS_EXPIRED           ,         Hostname was found, but it's status has expired. The address returned
                                             should not be used. Use resolv_query() to freshen it up.
   RESOLV_STATUS_NOT_FOUND         ,         The server has returned a not-found response for this domain name. This
                                             response is cached for the period described in the server. You may issue a
                                             new query at any time using resolv_query(), but you will generally want
                                             to wait until this domain's status becomes
                                             RESOLV_STATUS_EXPIRED.
   RESOLV_STATUS_RESOLVING         ,         This hostname is in the process of being resolved. Try again soon.
   RESOLV_STATUS_ERROR             ,         Some sort of server error was encountered while trying to look up this
                                             record. This response is cached and will eventually expire to
                                             RESOLV_STATUS_EXPIRED.

};
```

4.2.11 ule6lo_ipMode_t

Description: IP address mode.

C-syntax:

```
typedef enum ule6lo_ipMode_t
{
     IP_ADDRESS_ANY                = -1,       ANY
     IP_ADDRESS_TENTATIVE          = 0,        Tentative
     IP_ADDRESS_PREFERRED          = 1,        Preferred
     IP_ADDRESS_DEPRECATED         = 2         Deprecated
} ule6lo_ipMode_t;
```

5 Example on how the 6LoWPAN Library is interfaced