NGTFF

Version 2.0.0.0

M3UA
SCCP

Developer Guide

CONFIGURATION TAGS

TAG Name	Description
# Host Configuration	
HostAddressType	IPv4 if the M3UA AS/ASP needs to listen on IP v4 address IPv6 if the M3UA AS/ASP needs to listen on IP v6 address
HostAddress	IPv4 or IPv6 of the host machine Add another HostAddress for secondary HostAddress
HostPort	The Port of the host machine, if configured, the host will be listening to this address, if et al, required, another instance of AS/ASP, configure separate node
HostApplicationId	ApplicationId of Host
HostAspUpAckTimer	Default value is 2 for Timer in M3UA
HostAspTrafficModeType	1 Override 2 Loadshare 3 Broadcast
# ASP Configuration	
ASPId	int value
ASPLocalPointCode	Point Code of ASP
ASPRoutingContext	U16 value of RoutingContext, multiple RoutingContext can be configure.
ASPSGAddressType	Signaling Gateway IPAddress Type, IPv4 or IPv6
ASPSGPrimaryAddress	Signaling Gateway Primary Address
ASPSGSecondaryAddress	Signaling Gateway Secondary Address
ASPSGConnect	0 - will not connect to signaling gateway 1 - connects to signaling gateway
ASPSGPort	U16 - Signaling Gateway port, where ASP Needs to connect
ASPSGClientPort	U16 - When Connecting to Signaling Gateway, this port will be used to bind to local system

Multiple configuration entries of ASP are allowed.

Application Server/ Application Server Process - Development

Step 1: Initialize SS7 Stack
Step 2: Load Stack Configuration File
Step 3: Hook Message Callback function to SS7 Stack

in the main function or application initialization block, call be the below stack functions to initialize and setup stack.

```
__ngtff_ss7__init();
__ngtff_ss7__load_stackconfig( <stack config file>);
__ngtff_ss7__onrecvmsg( <function pointer to recv message>);
```

Receive Message

Receive Callback Function Signature

syntax:

```
void <function-name>( uint8_t * msg);
```

eg:

```
void asp_onrecvmsg( uint8_t * msg)
```

The Stack Calls the recv message callback function whenever it receives the M3UA SCCP Message from network,

The Messages will be encapsulated in *msg* parameter.

Protocol Data Information From M3UA

The below function is to provide protocol information from M3UA Layer.

```
uint32_t aspId;
uint32_t opc;
uint32_t dpc;
uint8_t ni;
uint8_t mp;
uint8_t sls;

__ngtff_ss7__get_protocol_data( msg, &aspId, &opc, &dpc, &ni, &mp, &sls);
```

Decode SCCP

```
NGTFF_SM sccpMsg;
memset( &sccpMsg, 0, sizeof( NGTFF_SM));

int decode_sts = __ngtff_ss7__decode_sccp( msg, &sccpMsg);

if( decode_sts != 1)
{
      //decoding failed
      return;
}
```

Decode SCCP Address

```
NGTFFSA calledPartyAddress;
memset( &calledPartyAddress, 0, sizeof( NGTFFSA));

NGTFFSA callingPartyAddress;
memset( &callingPartyAddress, 0, sizeof( NGTFFSA));

decode_sts = __ngtff_ss7__sccp_decode_address( &calledPartyAddress,
sccpMsg.calledPartyAddress , sccpMsg.calledPartyAddresslen);

if( decode_sts == 1)
      asp_print_address( &calledPartyAddress);

decode_sts = __ngtff_ss7__sccp_decode_address( &callingPartyAddress,
sccpMsg.callingPartyAddress , sccpMsg.callingPartyAddresslen);

if( decode_sts == 1)
      asp_print_address( &calledPartyAddress);
```