LTE TOTALeNodeB OAM

User Guide

1222464 5.0

Contents

Figures

Tables

1 Preface

1.1 Objective

This document describes the procedure to setup, configure, and demonstrate a single call using the LTE TOTALeNodeB (TeNB) Operations, Administration, and Maintenance (OAM) components designed by Radisys. This product is also referred as TeNB OAM in the rest of the document.

1.2 Audience

Radisys assumes that the readers of this document are:

- Product Development team
- Product Line Management team
- Sales team
- Test or Validation team
- Program Management team
- Existing and potential customers

The readers must have an understanding of TeNB and its architecture.

1.3 Document Organization

This document contains the following sections.

Table-1: Document Organization

Section	Description
1. Preface	Provides the objective and release details.
2. Introduction	Provides an overview of the product, including the product description and features.
3. Functional Architecture of OAM Components	Describes the OAM components architecture.
4. Software Requirements	Describes software required for OAM components.
5. Features Supported	Describes the OAM features supported.
6. Open Source Components Used	Describes the OSS components used to create the OAM components.
7. Configuring IPSec	Describes the steps to configure IPSec.
8. Executing OAM Components	Describes the steps to execute OAM components.
9. Watchdog Application	Describes the Watchdog application and its usage.
10. CLI Usage and Commands	Describes the usage of Command Line Interface (CLI) and the commands.
11. Appendix-A: ACSLite Setup	Describes the ACSLite setup for HeMS.

Section	Description
12. Appendix-B: Integration with Open Source Components	Describes the steps to integrate Open Source components with TeNB OAM.
13. Appendix-C: FAQs	Lists the frequently asked OAM questions and solutions.
14. Appendix-D: FAQs	Lists the frequently asked IPSec questions and solutions.
15. References	Lists the reference documents.

1.4 Definitions and Acronyms

The abbreviations and acronyms used in this document are listed in Table-2.

Table-2: Abbreviations and Acronyms

Acronym	Description
AES	Advanced Encryption Standard
ANR	Automatic Neighbor Relations
APN	Access Point Network
App	Sample Application Layer
CL	Convergence Layer
CLI	Command Line Interface
CM	Configuration Mode
DES	Data Encryption Standard
DL	Downlink
DN	Distinguished Name
EARFCN	E-UTRA Absolute Radio Frequency Channel Number
eNB/eNodeB	E-UTRAN Node B
E-UTRAN	Evolved UTRAN
FDD	Frequency Division Duplex
FM	Fault Management
FSM	Finite State Machine
FTP	File Transfer Protocol
HeNB	Home eNodeB
HeMS	Home eNodeB Management System

HSS	Home Subscriber Server
IE	Information Element
IKEv2	Internet Key Exchange v2
IKE SA	Internet Key Exchange Security Association
IP	Internet Protocol
IPsec / IPSec	IP Security
KPI	Key Performance Indicator
L-ARM / LARM	Lower ARM
LTE	Long Term Evolution
MAC	Medium Access Control Protocol
MCC	Mobile Country Code
MIB	Master Information Block
MME	Mobile Management Entity
MNC	Mobile Network Code
NAS	Non-access Stratum
NMM	Network Monitor Mode
NTP	Network Time Protocol
OAM	Operation, Administration and Maintenance
OAM&P	Operations, Administration, Maintenance, and Provisioning
OTA	Over-The-Air
PCI	Physical Cell Identifier
PDCCH	Physical Downlink Control Channel
PDCP	Packet Data Convergence Protocol
PDN	Packet Data Network
PDSCH	Physical Downlink Shared Channel
PHY	Physical Layer
PM	Performance Management
PO	Post Office
PUSCH	Physical Uplink Shared Channel

P-GW/PDN-GW/PGW	PDN Gateway
PLMN	Public Land Mobile Network
RAC	Radio Admission Control
RAN	Radio Access Network
REM	Radio Environment Monitoring
RF	Radio Frequency
RLC	Radio Link Control Protocol
RNC	Radio Network Controller
RPC	Remote Procedure Call
RRC	Radio Resource Control Protocol
RRM	Radio Resource Management
RV	Redundancy Version
S1AP	S1 Application Protocol
SCTP	Stream Control Transmission Protocol
S-GW/SGW	Serving Gateway
SeGW	Security Gateway
SIB	System Information Block
SM	Stack Manager
SoC	System-on-a-Chip
SON	Self-Optimizing Networks
SSI	System Services Interface
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TDD	Time Division Duplex
TeNB / TOTALeNB	TOTALeNodeB
TR-069	CPE WAN Management Protocol, Broadband Forum Technical Report-069
TTI	Transmission Timing Interval
TUCL	TCP/UDP Convergence Layer

U-ARM / UARM	Upper ARM
U-Boot	Universal Boot Loader
UDP	User Datagram Protocol
UDP/IP	User Datagram Protocol / Internet Protocol
UE	User Equipment
UL	Uplink
ULPC	Uplink Power Control
UTRAN	Universal Terrestrial Radio Access Network
WDApp	Watchdog Application
X2AP	X2 Application Protocol
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

For a list of commonly used terms, refer to the Engineering Glossary at www.radisys.com/resources/wireless-glossary/

1.5 Release History

The following table lists the history of changes in successive revisions to this document.

Table-3: Release History

Version	Date	Description
5.0	November 30, 2015	LTE TOTALeNodeB Solution release, version EA 5.0
4.0	May 08, 2015	LTE TOTALeNodeB Solution release, version GA 4.0
3.0	June 14, 2014	LTE TOTALeNodeB Solution release, version GA 3.0
2.0	February 24, 2014	LTE TOTALeNodeB Solution release, version GA 2.0
1.3	October 16, 2013	LTE TOTALeNodeB release, version GA 1.1 for Broadcom SoC.
1.2	July 03, 2013	LTE TOTALeNodeB release, version 2.0 Alpha.
1.1	April 29, 2013	LTE TOTALeNodeB release, version 1.1
1.0	January 22, 2013	LTE TOTALeNodeB release, version 1.0

2 Introduction

2.1 Product Description

The LTE TeNB OAM components support the following features:

1. **Home eNodeB Management System (HeMS)**

   Home eNodeB initialization and registration.

2. **Watchdog Application**

   Check stalled processes on the Home eNodeB and trigger restart.

3. **Configuration Management**

   - Broadband Forum defined standard interfaces:
     - TR-069 data model between HeMS and HeNB.
     - TR-196, TR-181 and TR-262 data models.

4. **Performance Management**

   - Key Performance Indicator (KPI) counters for eNodeB application and configurations related to collection intervals, collection periods using TR-262 data model.
   - XML file format for reporting performance counter details.
   - Command Line Interface for reading performance counter details.

5. **Fault Management**

   HeMS fault reporting

6. **Log Management**

   - Logging/trace information writing to local files and console.

   The log files are stored at **/rsys/setup/trace/** directory.

   Each binary has its own log file.

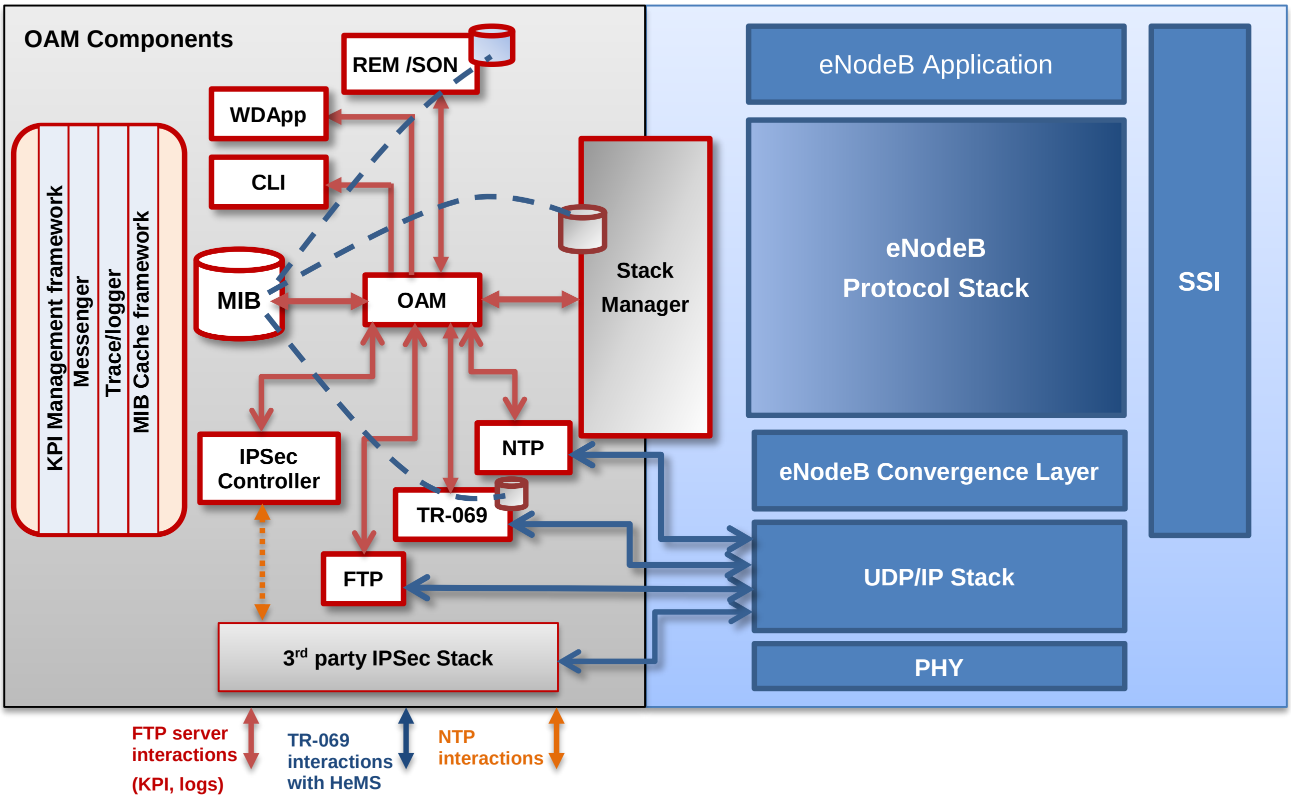
   For example:

     - post-office.21-05-2013_15-01-11.186680.txt
     - oam.21-05-2013_15-00-06.436284.txt
     - oam-sm.21-05-2013_15-20-18197518.txt
     - tr069.21-05-2013_15-01-30.805759.txt

3 Functional Architecture of OAM Components

The functional architecture of the OAM components is illustrated in Figure-1.

Figure-1: Functional Architecture of OAM Components



OAM components include:

- OAM framework and infrastructure components
- TR-069 client module
- CLI module
- IPSec controller module
- Network Time Protocol (NTP) client
- File Transfer Protocol (FTP) client
- Management Information Base (MIB) module (data model repository)
- REM controller module and SON module
- Watchdog Application (WDApp)

3.1 OAM Framework and Infrastructure Components

The following frameworks are the basic building blocks for OAM to perform its functions:

- **Messaging framework**: Provides a mechanism for inter-process communication between components (system applications) such as OAM, TR-069, CLI, FTP client and is termed as Post-Office (PO). The messenger is designed to provide UDP Post-Office like messaging service. Each of the platform application components are required to register with the messaging entity and each application is assigned specific ports to listen.
- **KPI management framework**: Provides mechanisms for building a list of supported Key Performance Indicators (KPIs).
- **Trace/logger**: Provides common interface for adding traceability and debugging capability for the components with a set of modules and macros.
- **MIB-Cache framework**: Maintains a consistent configuration data across the system components.
- **Start-up scripts**: Maintains scripts for environment settings, system startup, secure tunnel setup indication, bringing down application and system reboot.

3.2 TR-069 Client Module

The TR-069 Client provides the TR-069 protocol support interfacing the HeMS with a secure connection and helps exchange configuration and control information between HeMS and HeNB. Various data models are supported by the TR-069 stack and is mapped by the OAM component to the local configuration parameters and stored locally within HeNB.

3.3 Command Line Interface (CLI)

CLI module is developed in C++ to ease the set and get values function of the configuration parameters and provides an interface to use different OAM services. CLI sends commands to OAM through PO. When OAM receives CLI request from PO, OAM interacts with Master Information Block (MIB) or Radio Environment Monitoring (REM) to send the response to CLI through PO. Here, PO works as a messenger between CLI and OAM.

CLI sends command to TR-069 client module through PO. When TR-069 client module receives CLI command from PO, TR-069 client module sends the response to CLI through PO.

3.4 IP Security (IPSec) Controller Module

IPSec provides secure connection (tunnel mode) as required for securely exchanging control plane, user plane, TR-069 OAM messaging and NTP/FTP messages between HeNB and the core network infrastructure. IPSec management feature works closely with OAM and TR-069 modules.

3.5 Network Time Protocol (NTP) Client

NTP provides an alternative to Over-The-Air (OTA) synchronization with the macro cellular network. NTP client is responsible for interacting with the NTP server and implements NTP based frequency and system time synchronization algorithms. NTP client component interfaces with OAM using messaging interface.

3.6 File Transfer Protocol (FTP) Client

FTP client to be used for any kind of file transfer between eNodeB and HeMS through configured file transfer mechanism. Supported file transfer mechanisms are FTP, SFTP and SCP.

3.7 Management Information Base (MIB) Module (Data Model Repository)

The local storage of configuration and control parameters are designated as Management Information Base (MIB).

3.8 REM Controller and SON Module

Radio Environment Monitoring (REM) is required for location verification, Neighbor List (NL) configuration and parameter value selection (self-configuration). REM component includes the REM controller application and REM Convergence Layer (CL). REM controller interacts with OAM component using messaging framework for necessary configuration parameters and handles REM scan requests from OAM. REM CL is the interface to PHY and abstracts the PHY specifics from the controller. HeNB must support self-configuration and dynamic optimization. Self-Optimizing Network (SON) working along with REM controller provides basic and advanced features including self-configuration, self-optimization and self-healing.

3.9 Watchdog Application (WDApp)

The WDApp process uses UDP messaging and listens on UDP Port 6500. Third-party application must implement client UDP socket to send messages to WDApp and must start WDApp in the startup script.

The watchdog functionality includes the following components:

- WDApp: Monitors all registered applications.
- Kernel watchdog: Monitors the software WDApp.

The WDApp interacts with kernel watchdog to handle WDApp crash or stall.

4 Software Requirements

The OAM components are compatible on a 32-bit operating system. The following development tools must be installed to build and compile the OAM components:

- sun-java6 (-bin, -jre and -jdk)
- gcc, g++, gdb, binutils, cpp
- libcppunit (-1.x, -dev and -doc)
- libncurses5-dev
- php5-cli (required for fsmc)
- xsltproc (required for applying XSLT stylesheets to XML docs)
- Subversion (Tortoise SVN)

5 Features Supported

The following OAM features are supported.

Table-4: Features Supported

Feature	Support	Remarks
Static configuration	Yes	Parameters are described in the *TeNB_OAM_Supported_Parameters_API_Definition_1100105.xlsx* document.
Parameter validation	Yes	Parameters are described in the *TeNB_OAM_Supported_Parameters_API_Definition_1100105.xlsx* document.
First time bootup (BOOTSTRAP) procedure	Yes	Parameters for the first time initialization of the OAM.
BOOT procedure	Partial	Supports rebooting procedure of OAM and eNodeB components.
Dynamic updating	Partial	1. MME addition or modification or deletion 2. Neighbor cell 3. Neighbor eNB 4. Neighbor frequency 5. Measurement related parameters (Idle mode).
TR-069 procedures	Yes	1. Active or passive notification 2. Add or delete object 3. 6 CONNECTION REQUEST
Command Line Interface (CLI)	Yes	Parameters are described in the *TeNB_OAM_Supported_Parameters_API_Definition_1100105.xlsx* document.
Performance Management (PM)	Yes	L2 and L3 counters.
Fault Management (FM)	Yes	Supports fault generation.
Watchdog Application (WDApp)	Yes	A combination of software and kernel watchdog.

Refer to the *TeNB_OAM_Supported_Parameters_API_Definition_1100105.xlsx* document in the TeNB release package for more details.

6 Open Source Components

The following open source components are used for OAM module.

Table-5: Open Source Components

Reference Item	Reference Source	Objective of Usage	License	Comments
Boost C++	http://www.boost.org	Used by various components like OAM	http://www.boost.org/users/license.html	This is not part of the current delivery.
libCSOAP/nanoHTTP	http://csoap.sourceforge.net	Used by TR-069/TR-196 modules	GNU LGPLv2	This is not part of the current delivery.
libXML	http://xmlsoft.org/	Used by TR-069/TR-196 modules	MIT	This is not part of the current delivery.
OpenSSL	http://www.openssl.org	Version used is openssl-0.9.8r Used as a crypto library plugin for strongSwan (IPsec client)	OpenSSL and original SSLeay	This is not part of the current delivery.
Libgmp	http://gmplib.org	Version used is gmp-6.0.0a Required by strongSwan for compilation of strongSwan related binaries	GNU LGPLv2	This is not part of the current delivery.
strongSwan	https://www.strongswan.org	Version used is strongSwan-5.3.2 TeNB components tenpin and ike-tunnel-ind use strongSwan to establish the IPsec tunnel	GNU LGPLv2	This is not part of the current delivery.

The open source components must be configured and installed in

- SeGW machine
- Board (components to be cross compiled as per the SoC)

The SeGW machine used in the test setup is Ubuntu 12.04.

6.1 Pre-conditions

Following are the pre-conditions for the open source components usage:

1. Recommended to have the latest (compatible) **autoconf** version. However, this is not mandatory.
2. Execute the following command to set the compiler path in PATH environment variable:
   - ➔ **export PATH=/root/mipsel-unknown-linux-gnu/bin/:$PATH**
3. Execute the following command to give permission to all the folders:
   - ➔ **chmod 777 ***

6.2 Compilation Step

Go to the following path (**<TotalNodeB>/src/enbapp/build/**) and execute the **tpinstall** script.

Script can be used for compilation only, extraction only or both extraction and compilation.

1. For both extraction and compilation steps are
   - ➔ **cd <TotalNodeB>/src/enbapp/build/**
   - ➔ **./tpinstall.sh <COMPILER NAME>**

   For example: **./tpinstall.sh mipsel-unknown-linux-gnu**
2. For extraction only steps are
   - ➔ **cd <TotalNodeB>/src/enbapp/build/**
   - ➔ **./tpinstall.sh <COMPILER NAME> ext**

   For example: **./tpinstall.sh mipsel-unknown-linux-gnu ext**
3. For compilation only steps are
   - ➔ **cd <TotalNodeB>/src/enbapp/build/**
   - ➔ **./tpinstall.sh <COMPILER NAME> comp**

   For example: **./tpinstall.sh mipsel-unknown-linux-gnu comp**

This script updates the files in third-party software to make it compatible with TeNB software. Refer Appendix for details.

NOTE: Make sure root permission and net connectivity is there on compilation machine.

7 Configuring IPSec

This section describes the procedure to download the open source component required for IPSec and the configuration of the same.

IPSec secure connection is established between the LTE TeNB and the Security Gateway (SeGW) machine.

The SoC is also referred to as **target board or board** or **ipsec-client** in the rest of the document. Security Gateway machine is also referred to as **ipsec-server** in the rest of the document.

A virtual tunnel IP Address is assigned and configured on the board once the IPSec tunnel is established between the board and the SeGW machine.

7.1 Installation of Openssl

1. Download the **openssl-0.9.8r.tar.gz** file from http://www.openssl.org
2. Execute the following command to copy the downloaded file to **/opt** directory:

   ```
   cp openssl-0.9.8r.tar.gz /opt
   ```

3. Execute the following command to extract the copied file:

   ```
   cd /opt/
   tar -xvzf openssl-0.9.8r.tar.gz
   ```

4. Execute following command to rename **openssl-0.9.8r** folder to **origopenssl**

   ```
   mv –f /opt/openssl-0.9.8r/ <TotaleNodeB>/tenb_commonplatform/software/thirdparty/origopenssl
   cd <TotaleNodeB>/tenb_commonplatform/software/thirdparty/origopenssl
   cp –f <TotaleNodeB>/tenb_commonplatform/scripts/compilation_scripts/configs_openssl/configscript_openssl .
   find -exec dos2unix {} \;
   ```

5. Execute the following command to install the **openssl** package:

   ```
   ./configscript_openssl mipsel-unknown-linux-gnu
   ```

6. Execute the following command to copy **openssl** package to **/opt** folder:

   ```
   cp –f <TotaleNodeB>/tenb_commonplatform/software/thirdparty/origopenssl/../openssl-0.9.8/openssl-0.9.8rm/  /opt/ssl
   ```

7.2 Installation of Libgmp (required for strongSwan)

1. Download the **gmp-6.0.0a.tar.bz2** file from http://gmplib.org
2. Execute the following command to copy the downloaded file to **/opt** directory:

   ```
   cp gmp-6.0.0a.tar.bz2 /opt
   ```

3. Execute the following command to extract the copied file:

   ```
   tar -xvjf gmp-6.0.0a.tar.bz2
   cd gmp-6.0.0
   ```

4. Execute the following command to install the **gmp** package:

   ```
   ./configure --prefix=/opt/ --exec-prefix=/opt/ --host=mipsel-unknown-linux-gnu --build=mipsel CC=mipsel-unknown-linux-gnu-gcc CXX=mipsel-unknown-linux-gnu-g++ AR=mipsel-unknown-linux-gnu-ar LD=mipsel-unknown-linux-gnu-ld
   ```

```
AS=mipsel-unknown-linux-gnu-as RANLIB=mipsel-unknown-linux-gnu-ranlib --enable-shared --disable-static
```

```
make
```

```
make install
```

```
cp gmp.h  /opt/ssl/include/
```

```
cp .libs/* /opt/ssl/lib
```

7.3 Installation of strongSwan Version 5.3.2

1. Download the **strongswan-5.3.2.tar.bz2** file from http://download.strongswan.org/strongswan-5.3.2.tar.bz2
2. Execute the following command to copy the downloaded file to **/opt** directory:

   ```
   cp strongswan-5.3.2.tar.bz2 /opt
   ```

3. Execute the following command to extract the copied file:

   ```
   tar -xvjf strongswan-5.3.2.tar.bz2
   ```

4. Execute the following command to go to the **strongswan-5.3.2** directory:

   ```
   cd strongswan-5.3.2
   ```

5. Execute the following command to create **strongswan** directory under the **/opt** directory:

   ```
   mkdir /opt/strongswan
   ```

6. Execute the following command to copy the contents of **lib/*** directory to **/opt/strongswan** directory:

   ```
   cp /opt/ssl/lib/* /opt/strongswan
   ```

7. Execute the following commands to install the **strongswan-5.3.2** package:

   ```
   ./configure --enable-static --host=mipsel-unknown-linux-gnu --target=mipsel-unknown-linux-gnu --prefix=/opt/strongswan --libexecdir=/opt/strongswan/libexec --sysconfdir=/opt/strongswan/etc --enable-eap-aka --enable-eap-sim --enable-eap-sim-file --enable-eap-identity --enable-kernel-pfkey --enable-nat-transport --disable-pluto --disable-tools --enable-openssl --libdir=/opt/strongswan --includedir=/opt/ssl/include
   ```

   ```
   make
   ```

   ```
   make install
   ```

8. Go to the **/opt** directory and execute the **tar** command to compress the installed **strongswan** directory:

   ```
   cd /opt
   ```

   ```
   tar -cvf strongwan.tar strongswan
   ```

9. Execute the following command to copy the **strongswan.tar** file to the **target board** under the **/opt** directory:

   ```
   scp strongswan.tar root@<target board IP>:/opt/
   ```

10. Execute the following command in **target board** to extract the copied file:

    ```
    cd /opt
    ```

    ```
    tar –xvf strongswan.tar
    ```

7.4 Generation of Certificates

The step by step procedure to create sample digital certificates (x.509) used for mutual authentication of HeNB to SGW is as follows.

The certificates generated are self-signed. However, if there is a need to generate RootCA signed certificates refer the ‘openSSL’ documentation (www.openssl.org).

1. Execute the following command to create **certs** directory under the **/root** directory:

   ➔ **mkdir certs**

2. Execute the following command to go to the **certs** directory:

   ➔ **cd certs**

3. Execute the following commands to generate the certificates:

   ➔ **/opt/ssl/misc/CA.pl -newca**

   ➔ **/opt/ssl/misc/CA.pl -newreq**

   ➔ **/opt/ssl/misc/CA.pl -sign**

4. Execute the following commands to rename certificates:

   ➔ **mv newcert.pem carolcert.pem**

   ➔ **mv newreq.pem carolkey.pem**

   ➔ **mv newcacert.pem carolcacert.pem**

5. Execute the following command to change the permission of the certificates:

   ➔ **chmod 755 carolcert.pem carolkey.pem**

6. Execute the following commands to copy the certificates from build machine to the HeNB:

   ➔ **scp carolcert.pem root@<target board IP>:/opt/strongswan/etc/ipsec.d/certs**

   ➔ **scp carolkey.pem root@<target board IP>:/opt/strongswan/etc/ipsec.d/private**

7. Execute the following commands to edit the **ipsec.secrets** file and add the following line:

   ➔ **vi /opt/strongswan/etc/ipsec.secrets**

   ➔ **add :RSA carolkey.pem in the file**

Note: If the file does not exist, create on the target board.

8. Following parameters should be updated in the **<path>/config/configFile**:

   ➔ **STRONGSWAN_INSTALL_DIR**

   ➔ **STRONGSWAN_LEFTCERT_FILENAME**

   ➔ **STRONGSWAN_LEFT_ID**

9. Execute the steps (3 to 6) to create certificates for **ipsec-server**. Name the certificates as **moon** instead of **carol** and change the permission.

   ➔ **mv newcert.pem mooncert.pem**

   ➔ **mv newreq.pem moonkey.pem**

   ➔ **mv newcacert.pem mooncacert.pem**

   ➔ **chmod 755 mooncert.pem moonkey.pem mooncacert.pem**

10. Execute the following commands to copy the generated certificates on the **ipsec-server**.

    ➔ **cp mooncacert.pem /opt/strongswan/etc/ipsec.d/cacerts/**

    ➔ **cp mooncacert.pem /usr/local/etc/ipsec.d/cacerts/**

    ➔ **cp carolcacert.pem /usr/local/etc/ipsec.d/cacerts/**

11. The two certificates (**carolcacert.pem** and **mooncacert.pem**) of each machine (ipsec-server and ipsec-client), must be present on both ipsec-server and ipsec-client machines.

    ➔ **cp mooncert.pem /usr/local/etc/ipsec.d/certs/**

    ➔ **cp moonkey.pem /usr/local/etc/ipsec.d/private/**

Edit **/opt/strongswan/sbin/ipsec** in **ipsec-client** as follows:

- IPSEC_DIR="/opt/strongswan/libexec/ipsec"
- IPSEC_BINDIR="/opt/strongswan/bin"
- IPSEC_SBINDIR="/opt/strongswan/sbin"
- IPSEC_CONFDIR="/opt/strongswan/etc"

Ensure the following certificates and configuration files are present/copied and updated with parameters accordingly in **ipsec-client**:

- /opt/strongswan/etc/ipsec.secrets
- /opt/strongswan/etc/ipsec.conf
- /opt/strongswan/etc/ipsec.d/cacerts/carolcacert.pem
- /opt/strongswan/etc/ipsec.d/certs/carolcert.pem
- /opt/strongswan/etc/ipsec.d/private/carolkey.pem
- /opt/strongswan/sbin/ipsec

8 Executing OAM Components

The section describes the procedures to execute the OAM components.

8.1 Without IPSec

A single startup script (**start_TeNB)** executes OAM components and eNodeB. Before starting the script, appropriate configuration must be provided in the **configFile** placed at the following path.

<path>OAM/config/configFile	(for Broadcom platform)
<path>/rsys/config/configFile	(for Intel platform)

Go to the following path and execute the **start_TeNB** script:

- ➔ **cd <path>/OAM/setup/** (for Broadcom platform)
- ➔ **. ./start_TeNB**
- ➔ **cd <path>/rsys/setup/** (for Intel platform)
- ➔ **. start_TeNB**

8.2 With IPSec

A single startup script (**start_TeNB)** executes OAM components and eNodeB. Before starting the script, appropriate configuration must be provided in the **configFile** placed at the following path.

<path>OAM/config/configFile	(for Broadcom platform)
<path>/rsys/config/configFile	(for Intel platform)

Configure the following mandatory parameters in the **wr_cfg.txt** configuration file:

- STRONGSWAN_INSTALL_DIR
- STRONGSWAN_LEFTCERT_FILENAME
- STRONGSWAN_LEFT_ID
- IKE_IPSEC_ENABLE (IPSec: enable/disable “1/0”)
- IKE_IPSEC_GW_ADDR (IPSec SGW IP)
- SECUIRTY_GATEWAY_1 (SGW IP)
- IPSEC_SA_LIFETIME (Key generation time)
- IKE_SA_LIFETIME (Life time of IKE SA)
- IKE_DPD_INTERVAL (Dead peer detection interval)
- IKE_ENCRYPTION_NULL_ENABLE (Encryption algorithm: enable/disable “1/0”)
- IKE_ENCRYPTION_3DES_ENABLE (3DES encryption algorithm: enable/disable “1/0”)
- IKE_ENCRYPTION_AES_ENABLE (AES encryption algorithm: enable/disable “1/0”)
- IKE_ENCRYPTION_AES128_ENABLE (AES128 encryption algorithm: enable/disable “1/0”)
- MANAGEMENT_USERNAME (Username credential for HeMS connection)
- MANAGEMENT_PASSWORD (Password credential for HeMS connection)
- MANAGEMENT_SERVER (URL of HeMS).

Go to the following path and execute the **start_TeNB** script:

- ➔ **cd <path>/OAM/setup/** (for Broadcom platform)
- ➔ **. ./start_TeNB**
- ➔ **cd <path>/rsys/setup/** (for Intel platform)
- ➔ **. start_TeNB**

Note: Mandatory to configure PLMN (tr69.addobject Device.Services.FAPService.1.CellConfig.LTE.EPC.PLMNList) through CLI or HeMS to initialize eNodeB. The parameters can also be added in the configuration file.

Execute the following commands to configure PLMN from the CLI:

- ➔ **LTE_EPC_PLMN_ENABLE 1 FAP.0.LTE_CELL_PLMN_LIST.0**
- ➔ **LTE_OAM_PRIMARY_PLMN 1 FAP.0.LTE_CELL_PLMN_LIST.0**
- ➔ **LTE_OAM_PLMNID 22020 FAP.0.LTE_CELL_PLMN_LIST.0**

8.3 With IPSec (without using tenpin)

IPSec tunnel is established between **ipsec-server** and **ipsec-client** using **ipsec** utility.

1. Execute the following command in **ipsec-server** to start **ipsec** utility

- ➔ **ipsec start**

Note: Ensure that the certificate is valid with respect to the system date.

2. Execute the following command in **ipsec-client** to start **ipsec** utility

- ➔ **cd /opt/strongswan/sbin**
- ➔ **export LD_LIBRARY_PATH=/opt/strongswan/:.**
- ➔ **ipsec start**

Note 1: Strongswan package is mandatorily copied to **/opt** folder in Section 6.3

Note 2: Ensure that the certificate is valid with respect to the system date

3. Execute the following command in **ipsec-client** to retrieve manually the virtual IP Address created as part of the establishment of IPSec tunnel in the **ipsec-client**

- ➔ **ipsec statusall**

4. Edit **FGW0_IP_ADDRESS** parameter in **<path>OAM/config/configFile** in **ipsec-client** with the virtual IP Address obtained from the above command.

Note: Ensure appropriate configuration is done in the **configFile** to bring the TeNB up.

5. Go to the following path and execute the **start_TeNB** script:

- ➔ **cd <path>/OAM/setup/**
- ➔ **. ./start_TeNB**

Note: Mandatory to configure PLMN (tr69.addobject Device.Services.FAPService.1.CellConfig.LTE.EPC.PLMNList) through CLI or HeMS to initialize eNodeB. The parameters can also be added in the configuration file.

Execute the following commands to configure PLMN from the CLI:

- ➔ **LTE_EPC_PLMN_ENABLE 1 FAP.0.LTE_CELL_PLMN_LIST.0**
- ➔ **LTE_OAM_PRIMARY_PLMN 1 FAP.0.LTE_CELL_PLMN_LIST.0**
- ➔ **LTE_OAM_PLMNID 22020 FAP.0.LTE_CELL_PLMN_LIST.0**

9 Watchdog Application

The WDApp process uses UDP messaging and listens on UDP PORT 6500. Third-party application must implement client UDP socket to send messages to WDApp and must start WDApp in the startup script.

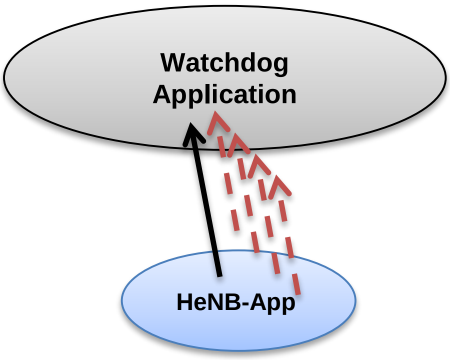
The watchdog functionality includes the following components:

- WDApp – Monitors all registered applications.
- Kernel watchdog – Monitors the software WDApp.

The WDApp interacts with kernel watchdog to handle WDApp crash or stall.

The functional architecture of the WDApp interacting with kernel watchdog is illustrated in Figure-2.

Figure-2: Functional Architecture of Watchdog Application



Example Scenario:

HeNB process monitoring through WDApp:

HeNB process requires to be monitored and gets registered with WDApp at startup and periodically sends heart beat message to WDApp to indicate that the process is active. When HeNB stops sending heart beat message, WDApp keeps count of successive missed heart beat messages and indicates the process is unresponsive, and triggers system reboot. Currently, WDApp counts three successive missed heart beat messages before recovering the process.

L-ARM monitoring through eNodeB application:

The eNodeB application monitors L-ARM through TTIs received from U-ARM.

The eNodeB application periodically heart beats the WDApp only if it continues to receive the TTIs.

The kernel watchdog monitors WDApp which is a software process running in user mode. The WDApp heart beats with kernel watchdog to indicate it is active. All the registered processes sends heart beat messages to WDApp to indicate that it is active.

9.1 Message Formats

The WDApp supports (or listens to) four message types:

1. WATCHDOG_ADD_PID_REQ

   This message is sent to WDApp by the process that requires monitoring. The message transmits process **ID** and **Descriptor** to describe the process (maximum length of 50 characters).

Table-6: WATCHDOG_ADD_PID_REQ

Data Type	Parameters
Header	srcEntity + destEntity + msgType + pid
Payload	appLen + appName

2. WATCHDOG_REMOVE_PID_REQ

This message is sent to WDApp by the process that does not require monitoring. The message can be sent by registered processes only and transmits process **ID**.

Table-7: WATCHDOG_REMOVE_PID_REQ

Data Type	Parameters
Header	srcEntity + destEntity + msgType + pid
Payload	---

3. WATCHDOG_KICK_REQ

This message is sent to WDApp by the process to indicate that it is alive. The message can be sent by registered processes only and transmits process **ID**, **softTimeout**, and **hardTimeout**.

softTimeout – The watchdog task reports a problem, if the context does not send heart beat messages within the given interval in seconds.

hardTimeout – The watchdog task stops sending heart beat messages to the hardware watchdog, if the context does not send heart beat messages within the given interval in seconds causing the system to reset.

Table-8: WATCHDOG_KICK_REQ

Data Type	Parameters
Header	srcEntity + destEntity + msgType + pid
Payload	softTimeout + hardTimeout + appLen + appName

4. WATCHDOG_KICK_STOP_REQ

This message is sent to WDApp by the process to stop heart beat message. The message can be sent by registered processes only and transmits process **ID**.

Table-9: WATCHDOG_KICK_STOP_REQ

Data Type	Parameters
Header	srcEntity + destEntity + msgType + pid
Payload	---

9.2 Logs

The watchdog events are printed on the console terminal. The WDApp logs can be redirected to a file (**logs.txt**) in the **watchdog&** directory by executing the following command:

```
./watchdog > logs.txt
```

The following logs are saved before attempting system recovery:

- RSYS L2 log
- MLOG
- Syscore log

9.3 Configurations

The following configuration must be done for WDApp:

1. Execute the following commands before running the WDApp to set soft limit and hard limit parameters using the **wr_cfg.txt** configuration file:

   ```
   cd rsys/bin/
   vi wr_cfg.txt
   ```

   Note: Set the following parameters only:

   WR_TAG_WATCHDOG_SOFT_LIMIT – **5** seconds (default value).

   WR_TAG_WATCHDOG_HARD_LIMIT – **15** seconds (default value).

2. Periodicity at which the WDApp monitors kick request from registered processes can be given as command line argument when running the WDApp. The recommended value is **5** seconds.

   Execute following command to set the values with WDApp:

   ```
   cd rsys/bin/
   . /watchdogd  <Watchdog Monitoring Time>  <Kernel Time-out> & > logs.txt
   ```

   Example-1: ./watchdogd 5 60 &
   Note: In this example, the values for <Watchdog Monitoring Time> and <Kernel Time-out> is assigned from command line.

   Example-2: ./watchdogd &
   Note: In this example, the values for <Watchdog Monitoring Time> and <Kernel Time-out> are taken from the **wr_cfg.txt** configuration file. The values (by default, is **5** seconds for Watchdog Monitoring Time and **15** seconds for Kernel Time-out) can be modified in the **wr_cfg.txt** configuration file as shown in step-1.

   Note: It is mandatory to start WDApp before eNodeB.

   Recommended Kernel Time-out value is 60 seconds (minimum time required to collect the logs and save it).

3. Path to save the log files using **wr_cfg.txt** configuration file:

4. Execute the following commands to set the parameters in the wr_cfg.txt file.

   ```
   cd  rsys/bin
   vi wr_cfg.txt
   ```

WR_TAG_LOG_PATH – **/var/log/** (default value).

9.4 Future Enhancements

The further enhancements for WDApp are as follows:

- Trace mechanism for the events that are logged to a file.
- Raise an alarm to notify OAM before system recovery to take appropriate decision or action based on the nature of the alarm.

10 CLI Usage and Commands

This section describes the CLI commands used for the OAM requirements.

1. Execute the following command to start the CLI application:

   ➔ **./cli**

2. Execute the following command to set the LTE parameters:

   ➔ **oam.set <parameter_name> <value> - Set the parameter**

3. Execute the following command to get the LTE parameters:

   ➔ **oam.getwild <parameter_name>**

   For example:

   Execute the following command to get the current admin state:

   ➔ **oam.getwild LTE_FAP_ADMIN_STATE**

   Execute the following command to get all the parameters:

   ➔ **oam.getwild LTE_***

4. Execute the following command to retrieve the performance counter values:

   ➔ **oam.pollkpis interval**

   Note: Counter values are redirected to OAM-sm* trace file under the **/opt/trace/** directory.

5. Execute the following command to retrieve the PM files:

   ➔ **tr69.gen.pm.file <filename>**

   Note: The file is generated in the directory where the CLI application is present.

The following mandatory list of parameters must be set through CLI:

1. LTE_BANDS_SUPPORTED
2. LTE_OAM_PLMNID
3. FGW0_IP_ADDRESS (not required when IPSec is enabled)
4. LTE_SIGLINK_SERVER_LIST
5. LTE_FREQUENCY_BAND_INDICATOR
6. LTE_SON_EARFCNDL_LIST
7. LTE_SON_EARFCNUL_LIST
8. LTE_CELL_IDENTITY
9. LTE_PHY_CELLID_LIST
10. FGW0_IP_ADDRESS
11. LTE_SMALLCELL_PCI_RANGE
12. LTE_SMALLCELL_START_PCI
13. LTE_CSG_PCI_RANGE
14. LTE_CSG_START_PCI
15. LTE_FAP_ADMIN_STATE

Add an object for PLMN (Device.Services.FAPService.1.CellConfig.LTE.EPC.PLMNList.) from CLI as follows:

1. LTE_EPC_PLMN_ENABLE 1 FAP.0.LTE_CELL_PLMN_LIST.0
2. LTE_OAM_PRIMARY_PLMN 1 FAP.0.LTE_CELL_PLMN_LIST.0
3. LTE_OAM_PLMNID 22020 FAP.0.LTE_CELL_PLMN_LIST.0

Note: For proper network operation, it is recommended that the CSG PCI range is configured as subset of small cell PCI range.

Refer to *TeNB_OAM_Supported_Parameters_API_Definition_1100105.xlsx* document in the TeNB release package for more details.

CLI commands are classified as four categories: CLI, MIB, OAM and TR-069.

Table-10: CLI Commands

Command	Sub-Command	Description
CLI	assert	Forces a state in the code to test assert mechanisms. For example: logging and watchdog.
	help	Displays help on CLI commands and usage.
	no-kick	Turns off auto shutdown after inactivity.
	showmem	Displays the memory usage by the CLI process.
	showtimers	Displays the timer usage by the CLI process.
MIB	assert	Forces a state in the code to test assert mechanisms. For example: logging and watchdog.
	create	Creates a new MIB object.
	delete	Deletes a MIB object (attempt fails if there are child objects).
	get	Gets a MIB attribute.
	get-desc	Gets a MIB attribute description.
	get-diffs-from-defaults	Gets all MIB attribute values that differ from the default values.
	get-nv	Gets all MIB attribute values that are explicitly set in NV file.
	get-subscriptions	Gets list of MIB subscriptions.
	getcellconfig	Gets key cell configuration MIB attributes.
	getwild	Gets a MIB attribute with wild card name search. For example: MIB getwild ENABLE
	set	Sets a MIB attribute. For example: MIB set SECURITY_GATEWAY_1 hms.secgw.com FAP.0.COMMISSIONING.0
	showmem	Displays the memory usage by the MIB process.

Command	Sub-Command	Description
	showtimers	Displays the timer usage by the MIB process.
OAM	alarms	Dumps all alarms to <file> or STDERR.
	assert	Forces a state in the code to test assert mechanisms. For example: logging and watchdog.
	Get	Gets a MIB attribute.
	get-ate-cli-version	Gets the CLI version.
	get-desc	Gets a MIB attribute description.
	Getwild	Gets a MIB attribute with wild card name search. For example: OAM getwild ENABLE
	mf.list	Lists all Managed Fings (MFs) that are registered including the messaging entity and current operation state.
	networking.ntp	Enables or disables NTP service.
	networking.restart	Restarts the networking service.
	Pollkpis	Triggers OAM to poll for all current KPI values without resetting them.
	rebootfap	Performs a FAP reboot.
	Scan	Triggers a REM scan request towards REM.
	sendalarm	Sends an alarm to OAM as if a real alarm event occurs. Must trigger all expected alarm event behavior.
	set	Sets a MIB attribute.
	showmem	Displays the memory usage by the OAM process.
	showtimers	Displays the timer usage by the OAM process.
	tracelev	Sets or gets trace criticality levels.
TR-069	action.factory.reset	Resets the MIB to factory settings.
	addobject	Adds multi instance object.
	alarms	Dumps all alarms to <file> or STDERR.
	assert	Forces a state in the code to test assert mechanisms. For example: logging and watchdog.
	clocks	Tests the clocks.
	ftpput	Copies a remote file to **/mnt/tmp** directory.

Command	Sub-Command	Description
	gen.pm.file	Generates a PM data file.
	gen.upload.pm.file	Generates a PM data file and uploads.
	showmem	Displays the memory usage by the TR-069 process.
	showtimers	Displays the timer usage by the TR-069 process.

10.1 CLI Commands

CLI.ASSERT

Forces a state in the code to test assert mechanisms. For example: logging and watchdog.

For example:
Command:

```
fap:/$ cli.assert
```

Output:

```
================================
!!!!!!!!!!! ASSERT !!!!!!!!!!!
Time: 05:14:59.808430
App: cli, generic, ver exported, 0 built at 12:18:29 01/10/2000
Thread: cli
Location: transport/CliHandler.cpp:574:CliCmdAssert()
Condition: Fail
Message: Assert forced through CLI
Errno: 0 (Success)
PID: 5571
Return: 0xb762d7d0 (cli) + ????: 0x8048000-0x80739d0)
Backtrace:
0: ./libthreeway-system.so(Trace_TraceAssert+0x64a) [0xb744d13e]
1: ./libthreeway-
messaging.so(_ZN8threeway10CliHandler12CliCmdAssertERKSt6vectorINS_11CliAr
gumentESaIS2_EE+0x7e) [0xb762ead4]
2: ./libthreeway-
messaging.so(_ZN8threeway10CliHandler13ExecuteCliCmdERKSsS2_+0x43e)
[0xb762d7d0]
..
.
Aborted
```

CLI.HELP

Displays help on CLI commands and usage.

For example:
Command:

```
fap:/$ cli.help
```

Output:

```
help - Show this text
help [namespaces | names | n] - List available namespaces
help [all | a] - List all available commands
help [commands | cmd | c] <partial-command-name> - Describe commands matching
<partial-command-name>
For example:  help c tr69
              help c tpm.show
```

```
                help c oam.netwo
OK
```

CLI.NO-KICK

Turns off auto shutdown after inactivity.

For example:
Command:

```
fap:/$ cli.no-kick
```

Output:

```
Done
OK
```

CLI.SHOWMEM

Displays the memory usage by the CLI process.

For example:
Command:

```
fap:/$ cli.showmem
```

Output:

```
{ Process Memory Summary
   MB Mem Used: 2.08203
   Total Program Size: 2132
   Resident Set Size: 1053
   Shared Pages: 920
   Text (Code): 44
   Library: 0
   Data / Stack: 95
   Dirty Pages: 0
}
OK
```

CLI.SHOWTIMERS

Displays the timer usage by the CLI process.

For example:
Command:

```
fap:/$ cli.showtimers
```

Output:

```
At: 19:29:33.240
Registered timers:
Handle:9, Name:CliCommandTimer, Period:30000, Single Shot
Total Registered timers: 1

Running timers:
At: 19:30:03.240 Handle: 9 Time to run: 30000 ms

Total Running timers: 1
OK
```

10.2 MIB Commands

MIB.ASSERT

Forces a state in the code to test assert mechanisms. For example: logging and watchdog.

For example:
Command:

```
fap:/$ mib.assert
```

Output:

```
================================
!!!!!!!!!!! ASSERT !!!!!!!!!!!
Time          : 05:41:29.127760
App           : oam, generic, ver unknown,0 built at 11:38:11 01/10/2000
Thread        : oam
Location :      transport/CliHandler.cpp:574:CliCmdAssert()
Condition:      Fail
Message       : Assert forced through CLI
Errno         : 2 (No such file or directory)
PID           : 5612
Return        : 0xb770e7d0 (oam) + ????: 0x8048000-0x818881b)
Backtrace:
   0:    ./libthreeway-system.so(Trace_TraceAssert+0x64a) [0xb741113e]
   1:    ./libthreeway-
         messaging.so(_ZN8threeway10CliHandler12CliCmdAssertERKSt6vectorINS_
         11C liArgumentESaIS2_EE+0x7e) [0xb770fad4]
   2:     ./libthreeway-
         messaging.so(_ZN8threeway10CliHandler13ExecuteCliCmdERKSsS2_+0x43
         e) [0xb770e7d0]

         .

         .

         .

         Time-out.
```

MIB.CREATE

Creates a new MIB object.

Usage: **mib.create <object-DN>**

For example:
Command:

```
fap:/$ mib.create FAP.0.FAP_LTE.0
```

Output:

```
MIB object created

OK
```

MIB.DELETE

Deletes a MIB object (attempt fails if there are child objects).

Usages: **mib.delete <object-DN>**

For example:

Command:

```
fap:/$ mib.delete FAP.0.FAP_LTE.0
```

Output:

```
MIB object deletion: MibObject deletion OK

OK
```

MIB.GET

Gets a MIB attribute.

Usages: **mib.get <attribute-name> [dn]**

For example-1:

Command:

```
fap:/$ mib.get LTE_BANDS_SUPPORTED FAP.0.FAP_LTE.0
LTE_BANDS_SUPPORTED
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,33,34,35,36,37,3 8,39,40
```

Output:

```
OK
```

For example-2:

Command:

```
fap:/$ mib.get LTE_SCTP_RTO_MAX LTE_SCTP_RTO_MAX 1000 (0x3e8)
```

Output:

```
OK
```

MIB.GET-DESC

Gets a MIB attribute description.

Usages: **mib.get-desc <attribute-name>**

For example:

Command:

```
fap:/$ mib.get-desc LTE_MAX_TX_POWER
```

Output:

```
LTE_MAX_TX_POWER 0 : default=0, minValue=0, maxValue=4294967295,
id=1551, moc=n, type=u32, access=R/O, storage=NonVolatile

OK
```

MIB.GET-DIFFS-FROM-DEFAULTS

Gets all MIB attribute values that differ from the default values.

For example:

Command:

```
fap:/$ mib.get-diffs-from-defaults
```

Output:

```
FAP.0: MNC=400 MCC=400 DL_UARFCN=10758 UMTS_BANDS_SUPPORTED="I"
.

.
.
OK
```

MIB.GET-NV

Gets all MIB attribute values that are explicitly set in NV file.

For example:
Command:

```
fap:/$ mib.get-nv
```

Output:

```
FAP.0: MNC=400 MCC=400 DL_UARFCN=10758 DL_FORCED_PSC=512 SAC=9
RNC_ID=96

.
.
.
OK
```

MIB.GET-SUBSCRIPTIONS

Gets list of MIB subscriptions.

For example:
Command:

```
fap:/$ mib.get-subscriptions
```

Output:

```
subscriber=4/OAM, subscriptionId=0, attr=[FAP.0: FAP_ADMIN_STATE], obj=

subscriber=4/OAM, subscriptionId=1, attr=[FAP.0:
NUMBER_UES_RRC_CONNECTED], obj=

subscriber=4/OAM, subscriptionId=2, attr=[FAP.0: FAP_ID UNIT_IP_ADDRESS],
obj=

subscriber=4/OAM, subscriptionId=3, attr=[FAP.0: DL_UARFCN], obj=
subscriber=4/OAM, subscriptionId=5, attr=[FAP.0: REQUIRE_FREQ_SYNC], obj=
subscriber=4/OAM, subscriptionId=8, attr=[FAP.0:
WCDMA_INTRA_FREQ_NEIGHBOUR_RNC_ID_1
WCDMA_INTRA_FREQ_NEIGHBOUR_RNC_ID_2

.

.

.

subscriber=9/OAM_HW, subscriptionId=7, attr=[FAP.0: DL_PRIM_SC], obj=
subscriber=9/OAM_HW, subscriptionId=8, attr=[FAP.0:
FREQ_SYNC_TIME_SYNCED PERIODIC_SCAN_INTERVAL_DAYS
PERIODIC_SCAN_ENABLED PERIODIC_SCAN_WINDOW_DUR_MINS
PERIODIC_SCAN_WINDOW_START_TOD_SECS], obj=

subscriber=9/OAM_HW, subscriptionId=9, attr=[FAP.0: DL_FORCED_PSC], obj=

OK
```

MIB.GETCELLCONFIG

Gets key cell configuration MIB attributes.

For example:
Command:

```
fap:/$ mib.getcellconfig
```

Output:

```
MNC = 400
MCC = 400
DL_UARFCN = 10758
SAC = 9
RNC_ID = 96
L2_CYPHERING_ENABLE = 1
SIB1_CS_DOMAIN_ENABLE = 1
SIB1_PS_DOMAIN_ENABLE = 1
SIB1_NMO = 1
CPICH_POWER_USED = -10
DL_PRIM_SC = 0
LAC = 8790
RAC = 90
CELL_IDENTITY = 50

OK
```

MIB.GETWILD

Gets a MIB attribute with wild card name search.

Usage: mib.getwild <pattern>

For example-1:
Command:

```
fap:/$ mib.getwild
```

Output:

```
HARDWARE_TYPE 5
HARDWARE_REVISION 1
HARDWARE_MOD_STATE 0
OSC_DAC 0

OSC_DAC_SLOPE_PPT_PER_BIT
5987 LONG_SERIAL_NUMBER
0911400199 MANUFACTURER
Radisys OUI_OF_MANUFACTURER
000050 GATEWAY_VENDOR RADISYS
DNS_SERVER_ADDRESS_1 0.0.0.0
DNS_SERVER_ADDRESS_2 0.0.0.0
DNS_SERVER_ADDRESS_3 0.0.0.0
MNC 400

MCC 400 DL_UARFCN 10758

DL_UARFCN_TO_PROTECT (NOT SET)
ALLOWED_DL_UARFCNS (NOT SET)

.
.
.
LTE_MAC_DBG 0
LTE_CL_DBG 0
LTE_SM_DBG 1
LTE_X2AP_DBG 0
LTE_HO_REPORT_CFG_VAL 1
LTE_INTER_FREQ_MEAS_GAP 1
LTE_ANR_MEAS_GAP_CONFIG 0
LTE_ANR_REPORT_CFG_VAL 1
LTE_INTRA_ANR_A3_OFFSET 5
LTE_INTER_ANR_A5_THRESHOLD1 75
```

```
LTE_INTER_ANR_A5_THRESHOLD2 50
OK
```

For example-2:
Command:

```
fap:/$ mib.getwild ENABLE
```

Output:

```
L2_CYPHERING_ENABLE 1
MEASUREMENT_LOGGER_ENABLE 0
INTRA_HO_ENABLE 1
INTER_HO_ENABLE 1
INTERRAT_HO_ENABLE 1
CS_VOICE_HO_ENABLE 1
CS_VIDEO_HO_ENABLE 1
.
.
LTE_SELFCONFIG_CELL_RESELECTION_ENABLE 0
LTE_SELFCONFIG_NEIGHBOUR_LIST_ENABLE 0
LTE_SELFCONFIG_PREAMBLE_RACH_TX_POWER_ENABLE 0
LTE_OAM_NEIGHBOUR_FREQ_UTRA_ENABLE 1
LTE_UL_ENABLE_TIME 10000

OK
```

MIB.SET

Sets a MIB attribute.

Usage: **mib.set <attribute-name> <attribute-value> [dn]**

For example-1:
Command:

```
fap:/$ mib.set SECURITY_GATEWAY_1
```

Output:

```
hms.secgw.com
OK
```

For example-2:
Command:

```
fap:/$ mib.set LTE_BANDS_SUPPORTED
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,33,34,35,36,37,3 8,39,40
```

Output:

```
FAP.0.FAP_LTE.0

OK
```

MIB.SHOWMEM

Displays the memory usage by the MIB process

For example:
Command:

```
fap:/$ mib.showmem
```

Output:

```
{ Process Memory Summary
   MB Mem Used: 2.08203
   Total Program Size: 2132
   Resident Set Size: 1053
   Shared Pages: 920
   Text (Code): 44
   Library: 0
   Data / Stack: 95
   Dirty Pages: 0
}
OK
```

MIB.SHOWTIMERS

Displays the timer usage by the MIB process.

For example:
Command:

```
fap:/$ mib.showtimers
```

Output:

```
At: 23:28:19.550 Registered timers:

Handle: 1, Name: Register Guard Timer, Period: 2000, Repeating
Handle: 2, Name: Subscribe Guard Timer, Period: 10000, Repeating
Handle: 3, Name: Ready Guard Timer, Period: 2000, Repeating
Handle: 4, Name: LED Not Operational timer, Period: 1800000, Single Shot w/o
Delete
Handle: 5, Name: LED No Connection to Server timer, Period: 30000, Single Shot
w/o Delete
Handle: 6, Name: OAMWatchdogKick, Period: 5000, Repeating Total Registered
timers: 6

Running timers:
At: 23:28:20.200 Handle: 6 Time to run: 650 ms

Total Running timers: 1

OK
```

10.3 OAM Commands

OAM.ALARMS

Dumps all alarms to <file> or STDERR.

Usages: **oam.alarms <file-name>**

For example-1:
Command:

```
fap:/$ oam.alarms
```

Output:

```
0 2000-10-01T05:11:19 ~Warning FAP Boot event FAP Boot Event (informative).
1 2000-10-01T05:16:59 ~Warning FAP Boot event FAP Boot Event (informative).
2 2000-10-01T06:14:05 ~Warning FAP Boot event FAP Boot Event (informative).
3 2000-10-01T05:13:29 ~Warning FAP Boot event FAP Boot Event (informative).
4 2000-10-01T05:12:26 ~Warning FAP Boot event FAP Boot Event (informative).
5 2000-10-01T05:37:25 ~Warning FAP Boot event FAP Boot Event (informative).
6 2000-10-01T05:15:18 ~Warning FAP Boot event FAP Boot Event (informative).

OK
```

For example-2:
Command:

```
fap:/$ oam.alarms /tmp/dd.log
```

Output:

```
OK
```

OAM.ASSERT

Forces a state in the code to test assert mechanisms. For example: logging and watchdog.

For example:
Command:

```
fap:/$ oam.assert
```

Output:

```
================================
!!!!!!!!!!!! ASSERT !!!!!!!!!!!!
Time          : 05:28:40.736561
App           : oam, generic, ver unknown,0 built at 11:38:11 01/10/2000
Thread        : oam
Location :      transport/CliHandler.cpp:574:CliCmdAssert()
Condition:      Fail
Message       : Assert forced through CLI
Errno         : 2 (No such file or directory)
PID           : 3678
Return        : 0xb765b7d0 (oam) + ????: 0x8048000-0x818881b)
Backtrace:
   0:  ./libthreeway-system.so(Trace_TraceAssert+0x64a) [0xb735e13e]
   1:  ./libthreeway-
       messaging.so(_ZN8threeway10CliHandler12CliCmdAssertERKSt6vectorINS
       _11C liArgumentESaIS2_EE+0x7e) [0xb765cad4]
   2:   ./libthreeway-
       messaging.so(_ZN8threeway10CliHandler13ExecuteCliCmdERKSsS2_+0x4
       3e) [0xb765b7d0]

       .
       .
       .
       Time-out.
```

OAM.GET

Gets a MIB attribute.

Usage: **oam.get <attribute-name>**

Note: Refer Table 2: List of attributes for detail information about attribute-name.

For example:
Command:

```
fap:/$ oam.get LTE_HO_REPORT_CFG_VAL
```

Output:

```
LTE_HO_REPORT_CFG_VAL 1 (0x1)

OK
```

OAM.GET-ATE-CLI-VERSION

Gets the CLI version.

For example:
Command:

```
fap:/$ oam.get-ate-cli-version 21
```

Output:

```
OK
```

OAM.GET-DESC

Gets a MIB attribute description.

Usage: **oam.get-desc <attribute-name>**

For example:
Command:

```
fap:/$ oam.get-desc  LTE_HO_REPORT_CFG_VAL
```

Output:

```
LTE_HO_REPORT_CFG_VAL 1: default=1, minValue=1, maxValue=2, id=1927,
moc=n, type=u32, access=R/W, storage=NonVolatile

OK
```

OAM.GETWILD

Gets a MIB attribute with wild card name search.

Usage: **oam.getwild <pattern>**

For example:
Command:

```
fap:/$ oam.getwild ENABLE
```

Output:

```
L2_CYPHERING_ENABLE 1
MEASUREMENT_LOGGER_ENABLE 0
INTRA_HO_ENABLE 1
INTER_HO_ENABLE 1
INTERRAT_HO_ENABLE 1
CS_VOICE_HO_ENABLE 1
CS_VIDEO_HO_ENABLE 1
PS_DATA_HO_ENABLE 0
SIB1_CS_DOMAIN_ENABLE 1
SIB1_PS_DOMAIN_ENABLE 1
ENABLE_TR069 1
.

.
.

LTE_SELFCONFIG_CELL_RESELECTION_ENABLE 0
LTE_SELFCONFIG_NEIGHBOUR_LIST_ENABLE 0
LTE_SELFCONFIG_PREAMBLE_RACH_TX_POWER_ENABLE 0
LTE_OAM_NEIGHBOUR_FREQ_UTRA_ENABLE 1 LTE_UL_ENABLE_TIME 10000

OK
```

OAM.MF.LIST

Lists all Managed Fings (MFs) that are registered including the messaging entity and current operation state.

For example:
Command:

```
fap:/$ oam.mf.list
```

Output:

```
MF_ETH0: Enabled "IP Interface Configured" (OAM_HW) OK
```

OAM.NETWORKING.NTP

Enables or disables NTP service.

Usage: **oam.networking.ntp <enable|disable>**

For example:
Command:

```
fap:/$ oam.networking.ntp enable
```

Output:

```
Enable

OK
```

OAM.NETWORKING.RESTART

Restarts the networking service.

For example:
Command:

```
fap:/$ oam.networking.restart
```

Output:

```
Networking should restart now...

OK
```

OAM.POLLKPIS

Triggers OAM to poll for all current KPI values without resetting them.

Usage: **oam.pollkpis <interval|total>**

For example-1:
Command:

```
fap:/$ oam.pollkpis interval
```

Output:

```
I've done it!  Now go check the oam trace file...

OK
```

For example-2:
Command:

```
fap:/$ oam.pollkpis total
```

Output:

```
I've done it!  Now go check the oam trace file...

OK
```

Note: you can find oam trace file under "trace” directory.

OAM.REBOOTFAP

Performs a FAP Reboot.

For example:
Command:

```
fap:/$ oam.rebootfap
```

Output:

```
Sent "Reboot Fap" to OAM. OK

Reboot requested by OAM fap:/$
```

```
Broadcast message from root@labadmin-ThinkCentre-M71e (/dev/pts/2) at 11:13 ...

The system is going down for reboot NOW!
```

Note: FAP has been restart after running this command and you may lose the working terminal.

OAM.SCAN

Triggers a REM scan request towards REM.

For example:
Command:

```
fap:/$ oam.scan
```

Output:

```
REM Scan requested...
OK
```

OAM.SENDALARM

Sends an alarm to OAM as if a real alarm event occurs. Must trigger all expected alarm event behavior.

Usage: **oam.sendalarm <id>**

[cleared|cl|warning|w|minor|mi|major|ma|critical|cr]
[transient|t|nontransient|n]
For example:
Command:

```
fap:/$ oam.sendalarm 12 w t
```

Output:

```
Sent alarm event to OAM: id=12(Over Temperature (Operational)), TRANSIENT,
severity=1, additionalInfo=Test alarm only!, alarmType=2, observationTime=2000-10-
01T05:25:32

OK
```

OAM.SET

Sets a MIB attribute.

Usage: **oam.set <attribute-name> <attribute-value>**

For example:
Command:

```
fap:/$ oam.set LTE_FAP_ADMIN_STATE 0
```

Output:

```
OK
```

OAM.SHOWMEM

Displays the memory usage by the OAM process.

For example:
Command:

```
fap:/$ oam.showmem
```

Output:

```
{ Process Memory Summary
    MB Mem Used                : 2.82227
    Total Program Size         : 2890
    Resident Set Size          : 1730
    Shared Pages               : 1427
    Text (Code)                : 321
    Library                    : 0
    Data / Stack               : 195
    Dirty Pages                : 0
}

OK
```

OAM.SHOWTIMERS

Displays the timer usage by the OAM process.

For example:
Command:

```
fap:/$ oam.showtimers
```

Output:

```
At: 00:12:07.650 Registered timers:
Handle: 1, Name: Register Guard Timer, Period: 2000, Repeating
Handle: 2, Name: Subscribe Guard Timer, Period: 10000, Repeating
Handle: 3, Name: Ready Guard Timer, Period: 2000, Repeating
Handle: 4, Name: LED Not Operational timer, Period: 1800000, Single Shot w/o
Delete
Handle: 5, Name: LED No Connection to Server timer, Period: 30000, Single Shot
w/o Delete
Handle: 6, Name: OAMWatchdogKick, Period: 5000, Repeating Total Registered
timers: 6

Running timers:
At: 00:12:09.090 Handle: 6 Time to run: 1440 ms

Total Running timers: 1

OK
```

OAM.TRACELEV

Sets or gets trace criticality levels.

For example:
Command:

```
fap:/$ oam.tracelev verbose on
```

Output:

```
Enabled Levels: VERBOSE INFO WARNING CRITICAL FATAL EXCEPTIONS
Available Levels: VERBOSE INFO WARNING CRITICAL FATAL CALL_STACK
EXCEPTIONS

Enabled Categories: [none] Available Categories: [none] OK
```

10.4 TR69 Commands

TR69.ACTION.FACTORY.RESET

Resets the MIB to factory settings.

For example:

Command:

```
fap:/$ tr69.action.factory.reset
```

Output:

```
Reboot requested by TR069

Attempting Factory Reset, (forces reboot!) OK

fap:/$
Broadcast message from root@labadmin-ThinkCentre-M71e (/dev/pts/1) at 11:26 ...

The system is going down for reboot NOW!
```

Note: FAP has been restart after running this command and you may lose the working terminal.

TR69.ADDOBJECT

Adds multi-instance object.

Usage: **tr69.addobject dn**

For example:
Command:

```
fap:/$ tr69.addobject
Device.Services.FAPService.1.CellConfig.LTE.RAN.NeighborList.LTECell.
```

Output:

```
Object created
```

When this command is issued a new instance for the specified multi-instance parameter is created with default values but not applied to application/layers. If at least one parameter of the created instance is SET, the complete object is passed to the application/layer.
In most cases, multi-instance objects have an “Enable” parameter. Unless this parameter is set to ‘1’ this object is not propagated to the application and hence this entry of object is not considered for processing.

TR69.DELETEOBJECT

Adds multi-instance object.

Usage: **tr69.deleteobject dn <instance number>**

For example:
Command:

```
fap:/$ tr69.deleteobject 1
Device.Services.FAPService.1.CellConfig.LTE.RAN.NeighborList.LTECell.
```

Output:

```
Object deleted
```

When this command is issued the instance specified in the command is no longer considered as a valid instance.

TR69.ALARMS

Dumps all alarms to <file> or STDERR.

Usage: **tr69.alarms <file-name>**

For example-1:
Command:

```
fap:/$ tr69.alarms
```

Output:

```
0 2000-10-01T05:11:19 ~Warning piLogFileMgr&] FAP Boot Event (informative).

1 2000-10-01T05:16:59 ~Warning piLogFileMgr&] FAP Boot Event (informative).
```

```
2 2000-10-01T06:14:05 ~Warning piLogFileMgr&] FAP Boot Event (informative).

3 2000-10-01T05:13:29 ~Warning piLogFileMgr&] FAP Boot Event (informative).

4 2000-10-01T05:12:26 ~Warning piLogFileMgr&] 05:39:36.356169
transport/AppMibAttributeCache.cpp:700 (dnIter->second.IsAttribute.

5 2000-10-01T05:12:26 ~Warning piLogFileMgr&] FAP Boot Event (informative).

6 2000-10-01T05:37:25 ~Warning piLogFileMgr&] FAP Boot Event (informative).

7 2000-10-01T05:15:18 ~Warning piLogFileMgr&] FAP Boot Event (informative).

8 2000-10-01T05:26:27 ~Warning piLogFileMgr&] 05:12:43.560005
transport/AppMibAttributeCache.cpp:700 (dnIter->second.IsAttribu~
9 2000-10-01T05:26:27 ~Warning piLogFileMgr&] 05:14:59.808430
transport/CliHandler.cpp:574 (Fail) CliCmdAssert(): Assert force~
10 2000-10-01T05:26:27 ~Warning piLogFileMgr&] 05:27:25.841331
transport/CliHandler.cpp:574 (Fail) CliCmdAssert(): Assert force~
11 2000-10-01T05:26:27 ~Warning piLogFileMgr&] 05:29:08.000322
transport/CliHandler.cpp:574 (Fail) CliCmdAssert(): Assert force~
12 2000-10-01T05:26:27 ~Warning piLogFileMgr&] 05:41:29.127760
transport/CliHandler.cpp:574 (Fail) CliCmdAssert(): Assert force~
13 2000-10-01T05:26:27 ~Warning piLogFileMgr&] 05:28:40.736561
transport/CliHandler.cpp:574 (Fail) CliCmdAssert(): Assert force~
14 2000-10-01T05:26:28 ~Warning piLogFileMgr&] FAP Boot Event (informative).
15 2000-10-01T05:11:27 ~Warning FAP Boot event FAP Boot Event (informative).

OK
```

For example-2:
Command:

```
fap:/$ tr69.alarms /tmp/tr69alarm.log
```

Output:

```
OK
```

TR69.ASSERT

Forces a state in the code to test assert mechanisms. For example: logging and watchdog.

For example:
Command:

```
fap:/$ tr69.assert
```

Output:

```
================================

!!!!!!!!!!!! ASSERT !!!!!!!!!!!!
Time            : 05:14:21.274276
App             : tr069, generic, ver exported,0 built at 12:08:19 01/10/2000
Thread          : tr069-v2
Location :        transport/CliHandler.cpp:574:CliCmdAssert()
Condition:        Fail
Message         : Assert forced through CLI
Errno           : 0 (Success)
PID             : 1774
Return          : 0xb77887d0 (tr069-v2) + ????: 0x8048000-0x846019c)
Backtrace:
```

```
0:   ./libthreeway-system.so(Trace_TraceAssert+0x64a) [0xb731313e]
1:   ./libthreeway-messaging.so(_ZN8threeway10CliHandler12CliCmdAssertERKSt6vectorINS_11C liArgumentESaIS2_EE+0x7e) [0xb7789ad4]
2:   ./libthreeway-messaging.so(_ZN8threeway10CliHandler13ExecuteCliCmdERKSsS2_+0x43e) [0xb77887d0]
```

TR69.CLOCKS

Tests the clocks.

For example:
Command:

```
fap:/$ tr69.clocks
```

Output:

```
TimeWrap: 2000-10-01T05:11:49 localtime: Sun Oct 1 11:11:49 2000 OK
```

TR69.FTPPUT

Copies a remote file to **/mnt/tmp** directory.

Usage: **tr69.ftpput username password filename**

For example:
Command:

```
fap:/$ tr69.ftpput root root123 tr69.pm.log.1
```

Output:

```
Upload requested...

OK
```

TR69.GEN.PM.FILE

Generates a PM data file

Usage: **tr69.gen.pm.file <file-name>**

For example:
Command:

```
fap:/$ tr69.gen.pm.file /tmp/tr69.pm.log
```

Output:

```
PM file generated.
OK
```

TR69.GEN.UPLOAD.PM.FILE

Generates a PM data file and uploads.

Usage: **tr69.gen.upload.pm.file <filename>**

For example:
Command:

```
fap:/$ tr69.gen.upload.pm.file tr69.pm.log
```

Output:

```
PM file generated and Upload requested...
OK
```

TR69.SHOWMEM

Displays the memory usage by the TR-069 process.

For example:
Command:

```
fap:/$ tr69.showmem
```

Output:

```
{ Process Memory Summary
    MB Mem Used                  : 10.3359
    Total Program Size           : 10584
    Resident Set Size            : 2345
    Shared Pages                 : 1510
    Text (Code)                  : 1049
    Library                      : 0
    Data / Stack                 : 7160
    Dirty Pages                  : 0
}

OK
```

TR69.SHOWTIMERS

Displays the timer usage by the TR-069 process.

For example:
Command:

```
fap:/$ tr69.showtimers
```

Output:

```
At: 00:15:20.510 Registered timers:
Handle: 1, Name: Tr069WatchdogKick, Period: 5000, Repeating
Handle: 5, Name: Start Rescan-stalled protection timer (15mins), Period: 900000,
Single Shot
Total Registered timers: 2
Running timers:
At: 00:15:23.170 Handle: 1 Time to run: 2660 ms
At: 00:21:04.880 Handle: 5 Time to run: 344370 ms
Total Running timers: 2
OK
```

10.5 Adding and Configuring Multi-instance Object

1. Add an Object by creating an instance using **tr69.addobject** command.

   For example:

   tr69.addobject Device.Services.FAPService.1.CellConfig. LTE.RAN.NeighborList.LTECell

2. Set required parameters that must be configured along with its MIB DN and instance number. For the parameters that are not configured default values shall be applied. For MIB DN, refer *TeNB_OAM_Supported_Parameters_API_Definition_1100105.xls* document.

3. Set the enable parameter of the object to **1** if available. Unless this parameter is set all configurations on this object will not be applied.

   For example:

   oam.set LTE_NEIGH_LIST_LTE_CELL_ENABLE 1 FAP.0.LTE_RAN_NEIGH_LIST_LTE_CELL.0

11 Appendix – A: ACSLite Setup

ACSLite is used as a HeMS simulator for end-to-end testing. The following steps must be followed for ACSLite setup:

Setup FAP for HeMS:

1. Execute the following commands to start the CLI application and update the username, password and URL parameter values in the FAP for HeMS:

   ➔ **cd rsys/bin/**

   ➔ **./cli**

   ➔ **oam.set MANAGEMENT_USERNAME rsys**

   ➔ **oam.set MANAGEMENT_PASSWORD pass**

   ➔ **oam.set MANAGEMENT_SERVER URL**

Setup HeMS:

2. Login to **172.27.4.244:6** client through **vncviewer**.

   Enter the password as **lte@sqa1**

   Note: Password is **lte@sqa2** for **172.27.4.244:7** client; and **lte@sqa3** for **172.27.4.244:8** client.

3. Connect to the following URL from an internet browser:

   http://172.27.4.244/management/index.php

4. Enter the following user name and password to log in:

   User Name: **lte.sqa1**

   Password: **lte@sqa1**

   Note 1: Also, **lte.sqa2** can be used as the user name with password **lte@sqa2**.

   Note 2: The 172.27.4.244 client must be ping-able from the eNodeB (add required routes at eNodeB and ACSLite).

5. Enter the username and password in the **Managed Devices** tab in the ACSLite application:

6. Select **BASIC** in the **HTTP Authentication Method** tab.

   Note: The **HTTPS Authentication Method** must be left as **DEFAULT**.

   Refer to the *ACSLite_Users_Guide_v2.0.0.0.doc* in the release package for more details.

Note: ACSLite is used as a simulator for HeMS, by Radisys internally.

12 Appendix – B: Integration with Open Source Components

Perform the following changes in the files to make them compatible with TeNB OAM:

a. **MD5**

a. **md5/md5.c**

Delete the following line:

```
#include "tools.h"
#include <config.h> (Only for ARM compiler)
```

Replace all **`u_char`** with **`uint8_t`** across the file.

In Function MDsign:

Replace

```
memdup(&newdata, data, len);
```

With

```
newdata = alloca(len);
memcpy(newdata,  data, len);
```

b. **md5/md5.h**

Include the following file:

```
#include <stdint.h>
```

Replace all **`u_char`** with **`uint8_t`** across the file.

b. **Kb_getc**

a. **kb_getc/kb_getc.c**

Modify the header

```
#include "../include/kb_getc.h"
```

To

```
#include "kb_getc.h"
```

Move the following lines from **kb_getc.h** file to **kb_getc.c** file:

```
extern int errno;
static struct termios termattr, save_termattr;
static int ttysavefd = -1;
static enum
{
  RESET, RAW, CBREAK
} ttystate = RESET;
```

c. **libsoap**

a. **csoap/nanohttp/nanohttp-logging.h**

Remove ONLY the following 3 lines of code at the end of the file

```
#ifdef __cplusplus

}

#endif
```

b. **csoap/nanohttp/nanohttp-client.c**

Add null check for socket. Open the socket only if it is free.

```
/* Open connection - only if needed. */

if(conn->sock.sock == HSOCKET_FREE)

{

      ssl = url.protocol == PROTOCOL_HTTPS ? 1 : 0;


      if ((status = hsocket_open(&conn->sock, url.host,
      url.port, ssl)) != H_OK)

      return status;

}
```

c. **csoap/nanohttp/nanohttp-common.c**

Modify this file to support cookie based authentication.

Insert the following lines

```
#include "nanohttp-common.h"

#include "nanohttp-logging.h"

char CookieValue_t[256];
```

Replace the **hpairnode_new** and **hpairnode_parse** functions with the following:

```
hpair_t *

hpairnode_new(const char *key, const char *value, hpair_t
* next)

{

   hpair_t *pair;

   size_t valuelength = 0;

   log_verbose3("new pair ('%s','%s')", SAVE_STR(key),
   SAVE_STR(value));

   pair = (hpair_t *) malloc(sizeof(hpair_t));


   if (key != NULL)

   {

      pair->key = (char *) malloc(strlen(key) + 1);
```

```
        strcpy(pair->key, key);
    }
    else
    {
        pair->key = NULL;
    }

    if (value != NULL)
    {
    if (key != NULL)
    {
        if(!strcmp(pair->key,"Cookie"))
        {
            pair->value = (char *)
            malloc(strlen(CookieValue_t) + 1);
            strcpy(pair->value, CookieValue_t);
        }
        else
        {
            pair->value = (char *) malloc(strlen(value) +
            1);
            strcpy(pair->value, value);
        }
    }
    }
    else
    {
        pair->value = NULL;
    }

    if(!strcmp(pair->key,"Cookie"))
    {
        strcpy(pair->value,CookieValue_t);
    }

    pair->next = next;

    return pair;
}
```

```
hpair_t *
hpairnode_parse(const char *str, const char *delim,
hpair_t * next)
{
   hpair_t *pair;
   char *key, *value;
   int c;

   pair = (hpair_t *) malloc(sizeof(hpair_t));
   pair->key = "";
   pair->value = "";
   pair->next = next;

   key = strtok_r((char *) str, delim, &value);

   if (key != NULL)
   {
      pair->key = (char *) malloc(strlen(key) + 1);
      strcpy(pair->key, key);
   }
   if (value != NULL)
   {
      for (c = 0; value[c] == ' '; c++);  /* skip white
      space */
      pair->value = (char *) malloc(strlen(&value[c]) +
      1);
      strcpy(pair->value, &value[c]);
      if(pair->key != NULL)
      {
         if(!strcmp(pair->key,"Set-Cookie"))
         {
            strcpy(CookieValue_t,&value[c]);
         }
      }
   }
   return pair;
}
```

d. **csoap/nanohttp/nanohttp-common.h**

Modification to take configured IP address for socket instead for INADDR_ANY.

Add the following hash define:

```
#define NHTTPD_ARG_ADDRESS     "-NHTTPaddress"
```

e. **csoap/nanohttp/nanohttp-server.c**

Modification to take configured IP address for socket instead of INADDR_ANY.

Define static variables as follows:

```
static uint32_t _httpd_address = INADDR_ANY;
static int _httpd_tos = 0;
```

Include if check in **`_httpd_parse_arguments`**:

```
else if (!strcmp(argv[i - 1], NHTTPD_ARG_ADDRESS))
   {
      sscanf(argv[i],"%lu", &_httpd_address);
   } else if (!strcmp(argv[i - 1], "TOS"))
      {
         _httpd_tos = atoi(argv[i]);
      }
```

At **`httpd_init`**, replace the following line

```
return hsocket_bind(&_httpd_socket, _httpd_port);
```

With

```
if(_httpd_tos != 0)
{
      return hsocket_bind_with_tos(&_httpd_socket,
      _httpd_address, _httpd_port,_httpd_tos);
}
   else
   {
      return hsocket_bind(&_httpd_socket, _httpd_address,
      _httpd_port);
  }
```

Add the following function to get the IP address:

```
uint32_t
httpd_get_address(void)
{
   return _httpd_address;
}
```

f. **csoap/nanohttp/nanohttp-socket.c**

Modification to take configured IP address for socket instead of INADDR_ANY.

Add the following hash define:

```
#define QOS_VALUE 20 //OAM
```

Modify function signature of **hsocket_bind** to accept IP address:

From

```
herror_t hsocket_bind(hsocket_t * dsock, int port)
```

To

```
hsocket_bind(hsocket_t * dsock, uint32_t address, int port)
```

To use the address while binding the socket:

Replace

```
addr.sin_addr.s_addr = INADDR_ANY;
```

With

```
addr.sin_addr.s_addr = htonl(address);
```

Add the following function:

```
/*----------------------------------------------------
FUNCTION: hsocket_bind
----------------------------------------------------*/
herror_t
hsocket_bind_with_tos(hsocket_t * dsock, uint32_t address, int
port, int tos)
{
   hsocket_t sock;
   struct sockaddr_in addr;
   int opt = 1;

   /* create socket */
   if ((sock.sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
   {
      log_error2("Cannot create socket (%s)", strerror(errno));
      return herror_new("hsocket_bind", HSOCKET_ERROR_CREATE,
      "Socket error (%s)", strerror(errno));
   }

   int retVal = setsockopt(sock.sock, SOL_SOCKET, SO_REUSEADDR,
   &opt, sizeof(opt));
```

```
        if ( tos == 0 )
        {
            tos = QOS_VALUE; //QOS_CLASS_E;
        }
        if(setsockopt(sock.sock, SOL_IP, IP_TOS, &tos,
        ((socklen_t)sizeof(tos)) ) != 0)
        {
            log_error2("Error setting setsockopt for TOS (%s)",
            strerror(errno));
            return herror_new("hsocket_bind", HSOCKET_ERROR_CREATE,
            "Socket options error (%s)", strerror(errno));
        }

        /* bind socket */
        addr.sin_family = AF_INET;
        addr.sin_port = htons((unsigned short) port); /* short,
        network byte order */
        addr.sin_addr.s_addr = htonl(address);
        memset(&(addr.sin_zero), '\0', 8);    /* zero the rest of
        the struct */

        if (bind(sock.sock, (struct sockaddr *) &addr, sizeof(struct
        sockaddr)) == -1)
        {
            log_error2("Cannot bind socket (%s)", strerror(errno));
            return herror_new("hsocket_bind", HSOCKET_ERROR_BIND,
            "Socket error (%s)", strerror(errno));
        }
        dsock->sock = sock.sock;
        return H_OK;
    }
```

Add the following lines in **_hsocket_sys_accept** function before returning **H_OK** to set socket options:

```
    int tos = QOS_VALUE; //QOS_CLASS_E;
        if(setsockopt(dest->sock, SOL_IP, IP_TOS, &tos,
        ((socklen_t)sizeof(tos)) ) != 0)
        {
            log_warn2("nanohttp-socket: setsockopt on accept failed
            (%s)", strerror(errno));
        }

        if(setsockopt(sock->sock, SOL_IP, IP_TOS, &tos,
        ((socklen_t)sizeof(tos)) ) != 0)
```

```
{
    log_warn2("nanohttp-socket: setsockopt on accept failed
    (%s)", strerror(errno));
}
```

g. **csoap/nanohttp/nanohttp-socket.h**

Modification to take configured IP address for socket instead of INADDR_ANY.

Replace the following line

```
herror_t hsocket_bind(hsocket_t * sock, int port);
```

With

```
herror_t hsocket_bind(hsocket_t * sock, uint32_t address,
int port);
```

Add the following function:

```
/**
    Binds a socket to the given port number. After bind, call
    the hsocket_listen() function to listen to the port as
    per hsocket_bind function, but with additional QoS-ToS
    parameter.

    @param sock: socket to use.

    @param port: port number to bind to

    @param tos: type of service for QoS socket options

    @returns H_OK value if success or one of the following
    values if failure:

        HSOCKET_ERROR_CREATE

        HSOCKET_ERROR_BIND

    @see hsocket_listen
*/

herror_t hsocket_bind_with_tos(hsocket_t * sock, uint32_t
address, int port, int tos);
```

13 Appendix – C: FAQs

13.1 TOTALeNodeB OAM FAQs

Q1: What does OAM component do?

Answer: OAM component helps in the configuring of the eNodeB, collecting and forwarding the Fault Management data and Key Performance Indicator parameters to the HeMS.

Q2: How is the communication set up between OAM and the HeMS configured?

Answer: The IP address of the HeMS server is configured in the MIB configuration file to be used by OAM to establish communication.

Q3: What kind of parameter configurations are supported from OAM?

Answer: Parameters are configured either statically or dynamically. The dynamically configurable list of parameters is mentioned in the *TeNB_OAM_Supported_Parameters_API_Definition_1100105.xlsx* document.

Q4: How is logging enabled or configured in OAM?

Answer: Logging is set or configured through the Command Line Interface for OAM. Logging for layers like RRC, eNBApp, MAC can be either enabled or disabled.

Q5: Where are the logs generated?

Answer: The log files are stored at the **<path>/setup/trace/** directory. Each binary has its own log file.

For example:
post-office.21-05-2013_15-01-11.186680.txt,
oam.21-05-2013_15-00-06.436284.txt,
oam-sm.21-05-2013_15-20-16.997518.txt,
tr069.21-05-2013_15-01-30.805759.txt

Q6: What is the maximum file size set for the OAM trace logs?

Answer: The maximum file size for OAM trace logs is configurable in the nas-system-configuration file. The default value configured is 2621440 KB.

Q7: Where are the KPI counters generated?

Answer: A file with KPI details gets generated at the **<path>/setup/trace** directory on demand by invoking the **oam.pollkpi interval** CLI command.

Another XML file with the counter values gets generated periodically at the same path as the binaries.

Q8: What is post office and why is it required?

Answer: Post Office (PO) convenes for inter process communication between system applications as OAM, TR-069, CLI and so on. The messenger is designed to provide UDP messaging service. Each application component is required to register with the messaging entity (PO) and each application is assigned specific port to listen.

Q9: Does OAM print any information on the eNodeB console?

Answer: All information provided by OAM is recorded in the log files and in the KPI measurement files.

Q10: Is the TR-069 client in OAM module tested for inter-operability with third-party TR-069 server (HeMS)?

Answer: Radisys OAM is configured and tested with the ACSLite HeMS server by NetMania. This testing includes remote configuration of eNodeB parameters through HeMS.

Q11: Is software upgrade of eNodeB supported through OAM?

Answer: This functionality is currently not supported.

Q12: Is configuration of SON supported through OAM?

Answer: SON configuration (ANR, REM) is supported.

Q13: Is there any documentation available for OAM?

Answer: The following documents are available for OAM:

i. TeNB_OAM_Integration_Guide_1555464.pdf

ii. TeNB_OAM_User_Guide_1222464.pdf

iii. TeNB_OAM_Supported_Parameters_API_Definition_1100105.xlsx

13.2 CLI Troubleshooting FAQs

Q.1: Why does the CLI stall when executing the **./cli** command?

Answer: The Post-Office is not functional on the same system. Verify with the following command:

➔ **ps –eaf|grep "post-office"**

If Post-Office is not functional, execute the following command to start the Post-Office:

➔ **./post-office &**

Q.2: Why is the following error displayed when executing the **./cli** command?

➔ **./cli**

"Detected that CLI is already running."

Answer: As mentioned in the error, another instance of CLI is running. Stop the existing instance to execute new CLI instance.

Q.3: Why is the TR-069 client unable to connect to the HeMS Server?

Answer: This issue is due to the username and password credentials. Please check section 11.1, step-5, on how to set the credentials through CLI.

14 Appendix – D: FAQs

14.1 IPSec FAQs

Q1: How to obtain IPSec logs for debugging IPSec tunnel establishment issues?

Answer: IPSec Strongswan related detailed logs can be obtained by adding below changes in **strongswan/etc/strongswan.conf** as follows:

```
charon {
    filelog {
         /var/log/charon.log {
         time_format = %b %e %T
         append = no
         default = 3
         flush_line = yes
          }
        }
    load_modular = yes
    plugins {
        include strongswan.d/charon/*.conf
    }
}
include strongswan.d/*.conf
```

Here strongswan logs will be present in /var/log/charon.log file. We can even provide the path /var/log/messages. These logs can be used for debugging any issues with IPSec tunnel establishment.

Q2: What is “Unmet dependency: NONCE_GEN” error log?

Answer: If you observe the below error message in **/var/log/messages** or in **/usr/local/etc/charon.log**, then it means that Strongswan is not properly linked and compiled.

Error Message:

```
uthpriv.info ipsec_starter[1116]: Starting strongSwan 5.3.2 IPsec [starter]...

aemon.info charon: 00[DMN] Starting IKE charon daemon (strongSwan 5.3.2, Linux
3.0.1brcm-0-1-rt11_CPUH_2_20, mi

aemon.info charon: 00[LIB] feature CUSTOM:libcharon in critical plugin 'charon' has unmet
dependency: NONCE_GEN

aemon.info charon: 00[LIB] feature CUSTOM:libcharon-receiver in critical plugin 'charon' has
unmet dependency: HASHER:HASH_SHA1

aemon.info charon: 00[LIB] failed to load 2 critical plugin features

aemon.info charon: 00[DMN] initialization failed - aborting charon

uthpriv.info ipsec_starter[1125]: charon has quit: initialization failed

uthpriv.info ipsec_starter[1125]: charon refused to be started

uthpriv.info ipsec_starter[1125]: ipsec starter stopped
```

Solution:
Compile the Strongswan package as mentioned in the steps provided in this document.

Q3: How to know IPSec tunnel establishment status?

Answer: To know the status of IPSEC Tunnel establishment status, use the following command:

ipsec statusall

Refer below sections in this document for example logs for IPSEC tunnel establishment.
If there is no tunnel established, then the above command result will be empty.

Q4: How to debug IPSec tunnel establishment failure issues?

Answer: If IPSec strongswan tunnel establishment fails in either the **ipsec-client** or in **ipsec-server**, then the following hints/tips will be helpful in debugging the issue:

1. Check if certificate is valid with the system time of **ipsec-server** and **ipsec-client**
2. Ensure that **"leftcert"** in **ipsec.conf** points to the correct **.pem** file path
3. Ensure that **"leftid"** in **ipsec.conf** points to the correct id as present in **.pem** file present in **"leftcert"**
4. Ensure that **"right"** in **ipsec-client's ipsec.conf** and **"left"** in **ipsec-server's ipsec.conf** contain the same value
5. Check Strongswan logs for detailed information

Q5: How to configure virtual tunnel IP address?

Answer: Virtual tunnel IP address will be created in the **ipsec-client** and configured during IPSec Tunnel establishment by the **ipsec-server**. This virtual tunnel IP address is used as TeNB IP Address for bringing up the cell.

The range of virtual tunnel IP addresses to be used is configured in **ipsec-server "rightsourceip"** parameter in **ipsec.conf** as shown below:

```
right=%any
rightsourceip=171.27.3.0/24
```

Q6: What to do if we observe Kernel Crash or Memory Page Fault in **ipsec-client**?

Answer: If Strongswan package is copied to **"/tmp"** folder and run in **ipsec-client**, then kernel crash or Memory page fault is observed when TeNB is run.

Solution:
Copy strongswan package to **"/opt"** folder and run. If the issue still persists contact respective Soc team.

Q7: How to configure NULL encryption for IPSec tunnel?

Answer: NULL encryption algorithm can be configured for the IPSec Tunnel to be established.

When configured for NULL encryption, debugging on IPSEC Tunnel packets is easier in Wireshark.

To configure NULL encryption, change **ipsec.conf** as follows in both **ipsec-client** and **ipsec-server**:

```
esp=null-sha1-modp1024!
```

Q8: How to check if certificate is valid?

Answer: Certificate validity is present in **.pem** file as follows:

```
Validity
        Not Before: Feb 20 16:39:25 2014 GMT
        Not After : Feb 20 16:39:25 2015 GMT
```

Hence system date should be within the range present in **.pem** file.

Q9: What is “modprobe: can't change directory” error?

Answer: The following error is observed when you run **“ipsec start”** command in **ipsec-client**:

```
./ipsec start
Starting strongSwan 5.3.2 IPsec [starter]...
modprobe: can't change directory to '3.0.1brcm-0-1-rt11_CPUH_2_22': No such file or directory
modprobe: can't change directory to '3.0.1brcm-0-1-rt11_CPUH_2_22': No such file or directory
modprobe: can't change directory to '3.0.1brcm-0-1-rt11_CPUH_2_22': No such file or directory
modprobe: can't change directory to '3.0.1brcm-0-1-rt11_CPUH_2_22': No such file or directory
modprobe: can't change directory to '3.0.1brcm-0-1-rt11_CPUH_2_22': No such file or directory
```

The above “No such file or directory” logs can be ignored.

Q10: How does successful IPv4 IPSec tunnel establishment log look like?

Answer: Sample IPv4 IPSec tunnel establishment is as follows:

```
./ipsec statusall
Status of IKE charon daemon (strongSwan 5.3.2, Linux 3.0.1brcm-0-1-rt11_CPUH_2_22, mips):
  uptime: 0 seconds, since Mar 18 14:54:10 2014
  malloc: sbrk 270336, mmap 0, used 190160, free 80176
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 6
  loaded plugins: charon aes des rc2 sha1 sha2 md5 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey pem openssl fips-prf gmp xcbc cmac hmac attr kernel-pfkey kernel-netlink resolve socket-default stroke updown eap-identity eap-sim eap-aka xauth-generic
Listening IP addresses:
  172.27.3.212
  1:2:3:4:5::10
  2000::10
Connections:
       conn1:  %any...172.27.3.218  IKEv2, dpddelay=6s
       conn1:   local:  [HNB1_svt@radisys.com] uses public key authentication
       conn1:    cert:  "C=IN, ST=KAR, O=Radisys India, OU=3G_SVT, CN=HNB1, E=HNB1_svt@radisys.com"
       conn1:   remote: uses public key authentication
       conn1:   child:  dynamic === 172.27.3.0/24 TUNNEL, dpdaction=clear
Security Associations (1 up, 0 connecting):
       conn1[1]: ESTABLISHED 0 seconds ago, 172.27.3.212[HNB1_svt@radisys.com]...172.27.3.218[SegGW1_svt@radisys.com]
       conn1[1]: IKEv2 SPIs: 208ec0c99bfd2248_i* e8708f44a78ac48c_r, rekeying in 23 hours, public key reauthentication in 23 hours
       conn1[1]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
       conn1{1}:  INSTALLED, TUNNEL, reqid 1, ESP SPIs: c74706dc_i c71c9ed9_o
```

```
    conn1{1}:  NULL/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 43 minutes
    conn1{1}:   171.27.3.1/32 === 172.27.3.0/24
```

Here 171.27.3.1 is the virtual IP created as part of establishing the tunnel in the board.

Q11: How does successful IPv6 IPSec tunnel establishment log look like?

Answer: Sample IPv6 IPSec tunnel establishment is as follows:

```
./ipsec statusall
Status of IKE charon daemon (strongSwan 5.3.2, Linux 3.0.1brcm-0-1-rt11_CPUH_2_20, mips):
  uptime: 4 seconds, since Mar 18 11:22:44 2014
  malloc: sbrk 270336, mmap 0, used 187648, free 82688
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 6
  loaded plugins: charon aes des rc2 sha1 sha2 md5 random nonce x509 revocation constraints pubkey pkcs1 pkcs7 pkcs8
pkcs12 pgp dnskey sshkey pem openssl fips-prf gmp xcbc cmac hmac attr kernel-pfkey kernel-netlink resolve socket-default
stroke updown eap-identity eap-sim eap-aka xauth-generic
Listening IP addresses:
  172.27.3.202
  2000::22
Connections:
       conn1:  %any6...2000::220  IKEv2, dpddelay=6s
       conn1:   local:  [HNB1_svt@radisys.com] uses public key authentication
       conn1:    cert:  "C=IN, ST=KAR, O=Radisys India, OU=3G_SVT, CN=HNB1, E=HNB1_svt@radisys.com"
       conn1:   remote: uses public key authentication
       conn1:   child:  dynamic === 2000::/64 TUNNEL, dpdaction=clear
Security Associations (1 up, 0 connecting):
       conn1[1]: ESTABLISHED 3 seconds ago, 2000::22[HNB1_svt@radisys.com]...2000::220[SegGW1_svt@radisys.com]
       conn1[1]: IKEv2 SPIs: 1800a65951aa67ed_i* e65c56beff7e8ae0_r, rekeying in 23 hours, public key reauthentication in 23
hours
       conn1[1]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
       conn1{1}:  INSTALLED, TUNNEL, reqid 1, ESP SPIs: c62bba20_i cb55c945_o
       conn1{1}:  NULL/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 46 minutes
       conn1{1}:   2001::1/128 === 2000::/64
```

Here 2001::1 is the virtual IP created as part of establishing the tunnel in the board.

15 References

Refer to the following documents for more information.

1. TeNB_REM_User_Guide_1222465.pdf
2. TeNB_BCM61750_FDD_User_Guide_1222603.pdf
3. TeNB_T2200_FDD_User_Guide_1222601.pdf
4. TeNB_OAM_Integration_Guide_1555464.pdf
5. TeNB_OAM_Supported_Parameters_API_Definition_1100105.xlsx
6. ACSLite_Users_Guide_v2.0.0.0.doc

radisys

www.radisys.com