|  |
| --- |
|  |
| rsys_logo |
| **LTE TOTALeNodeB OAM**  **User Guide**  **1222464 4.0** |
|  |
|  |
|  |

© 2015 by RadiSys Corporation. All rights reserved.

Radisys, Network Service-Ready Platform, Quick!Start, TAPA, Trillium, Trillium+plus, Trillium Digital Systems, Trillium On Board, TAPA, and the Trillium logo are trademarks or registered trademarks of RadiSys Corporation. All other trademarks, registered trademarks, service marks, and trade names are the property of their respective owners.

**Contents**

[1 Preface 7](#_Toc417668144)

[1.1 Objective 7](#_Toc417668145)

[1.2 Audience 7](#_Toc417668146)

[1.3 Document Organization 7](#_Toc417668147)

[1.4 Definitions and Acronyms 8](#_Toc417668148)

[1.5 Release History 11](#_Toc417668149)

[2 Introduction 12](#_Toc417668150)

[2.1 Product Description 12](#_Toc417668151)

[3 Functional Architecture of OAM Components 13](#_Toc417668152)

[3.1 OAM Framework and Infrastructure Components 14](#_Toc417668153)

[3.2 TR-069 Client Module 14](#_Toc417668154)

[3.3 Command Line Interface (CLI) 14](#_Toc417668155)

[3.4 IP Security (IPSec) Controller Module 14](#_Toc417668156)

[3.5 Network Time Protocol (NTP) Client 14](#_Toc417668157)

[3.6 File Transfer Protocol (FTP) Client 14](#_Toc417668158)

[3.7 Management Information Base (MIB) Module (Data Model Repository) 15](#_Toc417668159)

[3.8 REM Controller and SON Module 15](#_Toc417668160)

[3.9 Watchdog Application (WDApp) 15](#_Toc417668161)

[4 Software Requirements 16](#_Toc417668162)

[5 Features Supported 17](#_Toc417668163)

[6 Open Source Components 18](#_Toc417668164)

[6.1 Pre-conditions 18](#_Toc417668165)

[6.2 Compilation Step 19](#_Toc417668166)

[7 Configuring IPSec 20](#_Toc417668167)

[7.1 Installation of Libgmp (required for strongSwan) 20](#_Toc417668168)

[7.2 Installation of strongSwan Version 4.3.4 20](#_Toc417668169)

[7.3 Generation of Certificates 21](#_Toc417668170)

[8 Building TeNB with OAM Components 23](#_Toc417668171)

[8.1 Compile Options for Linux 23](#_Toc417668172)

[8.2 Compile Options for Broadcom FDD 23](#_Toc417668173)

[8.3 Compile Options for Broadcom TDD 23](#_Toc417668174)

[8.4 Compile Options for Intel T2200 24](#_Toc417668175)

[8.5 Compile Options for Intel T3300 24](#_Toc417668176)

[8.6 Build Packages 24](#_Toc417668177)

[9 Building OAM Components Separately 26](#_Toc417668178)

[10 Executing OAM Components 27](#_Toc417668179)

[10.1 Without IPSec 27](#_Toc417668180)

[10.2 With IPSec 27](#_Toc417668181)

[11 Watchdog Application 29](#_Toc417668182)

[11.1 Message Formats 30](#_Toc417668183)

[11.2 Logs 31](#_Toc417668184)

[11.3 Configurations 31](#_Toc417668185)

[11.4 Future Enhancements 31](#_Toc417668186)

[12 CLI Usage and Commands 32](#_Toc417668187)

[12.1 CLI Commands 35](#_Toc417668188)

[CLI.ASSERT 35](#_Toc417668189)

[CLI.HELP 35](#_Toc417668190)

[CLI.NO-KICK 36](#_Toc417668191)

[CLI.SHOWMEM 36](#_Toc417668192)

[CLI.SHOWTIMERS 36](#_Toc417668193)

[12.2 MIB Commands 37](#_Toc417668194)

[MIB.ASSERT 37](#_Toc417668195)

[MIB.CREATE 37](#_Toc417668196)

[MIB.DELETE 38](#_Toc417668197)

[MIB.GET 38](#_Toc417668198)

[MIB.GET-DESC 38](#_Toc417668199)

[MIB.GET-DIFFS-FROM-DEFAULTS 38](#_Toc417668200)

[MIB.GET-NV 39](#_Toc417668201)

[MIB.GET-SUBSCRIPTIONS 39](#_Toc417668202)

[MIB.GETCELLCONFIG 40](#_Toc417668203)

[MIB.GETWILD 40](#_Toc417668204)

[MIB.SET 41](#_Toc417668205)

[MIB.SHOWMEM 42](#_Toc417668206)

[MIB.SHOWTIMERS 42](#_Toc417668207)

[12.3 OAM Commands 43](#_Toc417668208)

[OAM.ALARMS 43](#_Toc417668209)

[OAM.ASSERT 43](#_Toc417668210)

[OAM.GET 44](#_Toc417668211)

[OAM.GET-ATE-CLI-VERSION 44](#_Toc417668212)

[OAM.GET-DESC 44](#_Toc417668213)

[OAM.GETWILD 44](#_Toc417668214)

[OAM.MF.LIST 45](#_Toc417668215)

[OAM.NETWORKING.NTP 45](#_Toc417668216)

[OAM.NETWORKING.RESTART 45](#_Toc417668217)

[OAM.POLLKPIS 46](#_Toc417668218)

[OAM.REBOOTFAP 46](#_Toc417668219)

[OAM.SCAN 46](#_Toc417668220)

[OAM.SENDALARM 47](#_Toc417668221)

[OAM.SET 47](#_Toc417668222)

[OAM.SHOWMEM 47](#_Toc417668223)

[OAM.SHOWTIMERS 48](#_Toc417668224)

[OAM.TRACELEV 48](#_Toc417668225)

[12.4 TR69 Commands 48](#_Toc417668226)

[TR69.ACTION.FACTORY.RESET 48](#_Toc417668227)

[TR69.ADDOBJECT 49](#_Toc417668228)

[TR69.DELETEOBJECT 49](#_Toc417668229)

[TR69.ALARMS 49](#_Toc417668230)

[TR69.ASSERT 50](#_Toc417668231)

[TR69.CLOCKS 51](#_Toc417668232)

[TR69.FTPPUT 51](#_Toc417668233)

[TR69.GEN.PM.FILE 51](#_Toc417668234)

[TR69.GEN.UPLOAD.PM.FILE 51](#_Toc417668235)

[TR69.SHOWMEM 51](#_Toc417668236)

[TR69.SHOWTIMERS 52](#_Toc417668237)

[12.5 Adding and Configuring Multi-instance Object 52](#_Toc417668238)

[13 Appendix – A: ACSLite Setup 53](#_Toc417668239)

[14 Appendix – B: Integration with Open Source Components 54](#_Toc417668240)

[15 Appendix – C: FAQs 62](#_Toc417668241)

[15.1 TOTALeNodeB OAM FAQs 62](#_Toc417668242)

[15.2 CLI Troubleshooting FAQs 63](#_Toc417668243)

[16 References 64](#_Toc417668244)

**Figures**

[Figure-1: Functional Architecture of OAM Components 13](#_Toc417668245)

[Figure-2: Functional Architecture of Watchdog Application 29](#_Toc417668246)

**Tables**

[Table-1: Document Organization 7](#_Toc417668247)

[Table-2: Abbreviations and Acronyms 8](#_Toc417668248)

[Table-3: Release History 11](#_Toc417668249)

[Table-4: Features Supported 17](#_Toc417668250)

[Table-5: Open Source Components 18](#_Toc417668251)

[Table-6: Compile Options for Linux Platform 23](#_Toc417668252)

[Table-7: Compile Options for Broadcom Platform 23](#_Toc417668253)

[Table-8: Compile Options for Broadcom Platform 23](#_Toc417668254)

[Table-9: Compile Options for Intel T2200 Platform 24](#_Toc417668255)

[Table-10: Compile Options for Intel T3300 Platform 24](#_Toc417668256)

[Table-11: WATCHDOG\_ADD\_PID\_REQ 30](#_Toc417668257)

[Table-12: WATCHDOG\_REMOVE\_PID\_REQ 30](#_Toc417668258)

[Table-13: WATCHDOG\_KICK\_REQ 30](#_Toc417668259)

[Table-14: WATCHDOG\_KICK\_STOP\_REQ 30](#_Toc417668260)

[Table-15: CLI Commands 33](#_Toc417668261)

# Preface

## 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.

## Audience

Radisys assumes that the readers of this document are:

* Product Development team,
* Product Line Management team,
* Sales team,
* Test/Validation team,
* Program Management team,
* Existing and potential customers.

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

## Document Organization

This document contains the following sections:

**Table-1: Document Organization**

|  |  |
| --- | --- |
| **Section** | **Description** |
| 1. Preface | Provides the objective and release details. |
| 1. Introduction | Provides an overview of the product, including the product description and features. |
| 1. Functional Architecture of OAM Components | Describes the OAM components architecture. |
| 1. Software Requirements | Describes software required for OAM components. |
| 1. Features Supported | Describes the OAM features supported. |
| 1. Open Source Components Used | Describes the OSS components used to create the OAM components. |
| 1. Configuring IPSec | Describes the steps to configure IPSec. |
| 1. Building TeNB with OAM Components | Describes the steps to build TeNB with OAM components. |
| 1. Building OAM Components Separately | Describes the steps to build OAM components individually. |
| 1. Executing OAM Components | Describes the steps to execute OAM components. |
| 1. Watchdog Application | Describes the Watchdog application and its usage. |
| 1. CLI Usage and Commands | Describes the usage of Command Line Interface (CLI) and the commands. |
| 1. Appendix-A: ACSLite Setup | Describes the ACSLite setup for HeMS. |
| 1. Appendix-B: Integration with Open Source Components | Describes the steps to integrate Open Source components with TeNB OAM. |
| 1. Appendix-C: FAQs | Lists the frequently asked OAM questions and solutions. |
| 1. References | Lists the reference documents. |

## 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 |
| 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/](mailto:STRONGSWAN_LEFT_ID=HNB1_svt@radisys.com)

## Release History

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

**Table-3: Release History**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Description** |
| 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 |

# Introduction

## Product Description

The LTE TeNB OAM components support the following features:

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

* Home eNodeB initialization and registration.

1. **Watchdog Application**

* Check stalled processes on the Home eNodeB and trigger restart.

1. **Configuration Management**

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

1. **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.

1. **Fault Management**

* HeMS fault reporting

1. **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

# Functional Architecture of OAM Components

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

Figure-1: Functional Architecture of OAM Components

**FTP server interactions**

**(KPI, logs)**

**NTP interactions**

**TR-069 interactions with HeMS**

**OAM**

**CLI**

eNodeB Application

**NTP**

**IPSec Controller**

**SSI**

**MIB**

**Messenger**

**KPI Management framework**

**Trace/logger**

**MIB Cache framework**

**Stack**

**Manager**

**TR-069**

**eNodeB**

**Protocol Stack**

**eNodeB Convergence Layer**

**UDP/IP Stack**

**OAM Components**

**PHY**

**REM /SON**

**FTP**

**3rd party IPSec Stack**

**WDApp**

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)

## 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.

## 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.

## 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.

## 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.

## 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.

## 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.

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

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

## 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.

## 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/stall.

# 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)

# Features Supported

The OAM features supported are as follows:

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/modification/deletion 2. Neighbor cell 3. Neighbor eNB 4. Neighbor frequency 5. Measurement related parameters (Idle mode). |
| TR-069 procedures | Yes | 1. Active/passive notification 2. Add/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.

# Open Source Components

The open source components used for OAM module are as follows.

Table-5: Open Source Components

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reference Item** | **Reference Source** | **Objective of Usage** | **License** | **Comments** |
| Boost C++ | [http://www.boost.org](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](http://www.openssl.org/source/openssl-0.9.8r.tar.gz) | 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](ftp://ftp.gnu.org/gnu/gmp/gmp-5.0.2.tar.bz2) | Required by strongSwan for compilation of strongSwan related binaries | GNU LGPLv2 | This is not part of the current delivery. |
| strongSwan | [www.](http://csoap.sourceforge.net/)[http://strongswan.org](http://download.strongswan.org/strongswan-4.3.4.tar.bz2)[.org](http://www.strongswan.org/) | Version used is strongSwan-4.3.4  TeNB components tenpin and ike-tunnel-ind use strongSwan to establish the IPsec tunnel | GNU LGPLv2 | This is not part of the current delivery. |

## Pre-conditions

The pre-condition for the open source components usage is as follows:

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**

1. Execute the following command to give permission to all the folders:

* **chmod 777 \***

## Compilation Step

Go to the following path and execute the **tpinstall** script with compiler name:

* **cd <TotalNodeB>/src/enbapp/build/**
* **./tpinstall.sh <COMPILER NAME>**

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

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

# Configuring IPSec

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

## Installation of Libgmp (required for strongSwan)

1. Download the **gmp-5.0.2.tar.bz2** file from [ftp://ftp.gnu.org/gnu/gmp/gmp-5.0.2.tar.bz2](http://www.strongswan.org/)
2. Execute the following command to copy the downloaded file to **/opt** directory:

* **cp gmp-5.0.2.tar.bz2 /opt**

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

* **tar -xvjf gmp-5.0.2.tar.bz2**
* **cd gmp-5.0.2**

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

* **./configure --prefix=/opt/ --exec-prefix=/opt/ --target=arm-none-linux-gnueabi --host=i686-pc-linux --build=arm CC=arm-none-linux-gnueabi-gcc CXX=arm-none-linux-gnueabi-g++ AR=arm-none-linux-gnueabi-ar LD=arm-none-linux-gnueabi-ld AS=arm-none-linux-gnueabi-as RANLIB=arm-none-linux-gnueabi-ranlib --enable-shared --disable-static**
* **make**
* **make install**
* **cp gmp.h /opt/ssl/include/**
* **cp .libs/\* /opt/ssl/lib**

## Installation of strongSwan Version 4.3.4

1. Download the **strongswan-4.3.4.tar.bz2** file from [http://download.strongswan.org/strongswan-4.3.4.tar.bz2](http://www.strongswan.org/)
2. Execute the following command to copy the downloaded file to **/opt** directory:

* **cp strongswan-4.3.4.tar.bz2 /opt**

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

* **tar -xvjf strongswan-4.3.4.tar.bz2**

1. Execute the following command to go to the **strongswan-4.3.4** directory:

* **cd strongswan-4.3.4**

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

* **mkdir /opt/strongswan**

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

* **cp /opt/ssl/lib/\* /opt/strongswan**

1. Execute the following commands to install the **strongswan-4.3.4** package:

* **./configure --enable-static --host=i686-pc-linux --target=arm-none-linux-gnueabi --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**

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

* **cd /opt**
* **tar -cvf strongwan.tar strongswan**

1. Execute the following command to copy the **strongswan.tar** file to the target board (HeNB) under the **/opt** directory:

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

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

* **tar –xvf strongswan.tar**

## 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](file:///C:/Users/rtripathy/AppData/Local/Microsoft/Windows/Temporary%20Internet%20Files/Content.Outlook/MDEWDQF8/www.openssl.org)).

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

* **mkdir certs**

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

* **cd certs**

1. 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**

1. Execute the following commands to rename certificates:

* **mv newcert.pem carolcert.pem**
* **mv newreq.pem carolkey.pem**
* **mv newcacert.pem carolcacert.pem**

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

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

1. 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**

1. 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.

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

* **STRONGSWAN\_INSTALL\_DIR**
* **STRONGSWAN\_LEFTCERT\_FILENAME**
* [**STRONGSWAN\_LEFT\_ID**](http://sourceforge.net/projects/boost/files/boost/1.52.0/boost_1_52_0.tar.gz/download)

1. 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**

1. 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/**

1. 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/**

# Building TeNB with OAM Components

This section describes the compilation and build of OAM components with Radisys TeNB.

## Compile Options for Linux

Go to **/enbapp/build** directory and execute one of the following commands as required.

Table-6: Compile Options for Linux Platform

|  |  |
| --- | --- |
| **Compilation Command** | **Description** |
| make lnxsplit | Linux compilation |
| make lnxsplit\_oam | Linux compilation with OAM feature enabled |
| make lnxclean | Clean object files |
| make lnxclean\_oam | Clean object files of OAM |

## Compile Options for Broadcom FDD

Go to **/enbapp/build** directory and execute one of the following commands as required.

Table-7: Compile Options for Broadcom Platform

|  |  |
| --- | --- |
| **Compilation Command** | **Description** |
| make bcfdd | Broadcom compilation for FDD |
| make bcfdd\_oam\_bld | MIPS (Broadcom) compilation with OAM feature enabled for FDD |
| make bccleanfdd\_oam\_bld | Clean object files |

## Compile Options for Broadcom TDD

Go to **/enbapp/build** directory and execute one of the following commands as required.

Table-8: Compile Options for Broadcom Platform

|  |  |
| --- | --- |
| **Compilation Command** | **Description** |
| make bctdd | Broadcom compilation for TDD |
| make bctdd\_oam\_bld | MIPS (Broadcom) compilation with OAM feature enabled for TDD |
| make bccleantdd\_oam\_bld | Clean object files |

## Compile Options for Intel T2200

Go to **/enbapp/build** directory and execute one of the following commands as required.

Table-9: Compile Options for Intel T2200 Platform

|  |  |
| --- | --- |
| **Compilation Command** | **Description** |
| make t2k\_int | T2200 compilation |
| make t2k\_oam  make t2k\_enb | T2200 compilation with OAM feature enabled |
| make t2k\_clean\_all | Clean object files |

## Compile Options for Intel T3300

Go to **/enbapp/build** directory and execute one of the following commands as required.

Table-10: Compile Options for Intel T3300 Platform

|  |  |
| --- | --- |
| **Compilation Command** | **Description** |
| make mse2e | ARM (Intel) compilation |
| make mse2e\_oam | ARM (Intel) compilation with OAM feature enabled |
| make msclean\_oam | Clean OAM object files |
| make msclean | Clean object files |

## Build Packages

After compilation, go to **/enbapp/build** directory and execute the following commands to build the package.

1. Execute the following command to build package in Linux platform:

* **./TeNBCP\_build linux**

When prompted, “Please enter the path where the build is to be copied”, enter the directory name as follows:

* **/root/enb/**

The build command creates the **/root/enb/** directory structure in the package:

**/root/enb/**

**|--->rsys**

1. Execute the following command to build package for Intel (T2K and T3K) platforms:

* **./TeNBCP\_build mspd**

When prompted, “Please enter the path where the build is to be copied”, enter the directory name as follows:

* **/root/enb/**

The build command creates the **/root/enb/** directory structure in the package:

**/root/enb/**

**|--->rsys/bin**

**|--->rsys/config**

**|--->rsys/lib**

**|--->rsys/nmmscripts**

**|--->rsys/setup**

1. Execute the following command to build package for Broadcom platform:

* **./TeNBCP\_build brcm**

When prompted, “Please enter the path where the build is to be copied”, enter the directory name as follows:

* **/root/enb/**

The build command creates the following directory structure in the package:

**/root/enb/**

**|--->OAM/bin**

**|--->OAM/config**

**|--->OAM/lib**

**|--->OAM/setup**

**Note**: Recommended to compress (tar) the package files and then copy or share.

# Building OAM Components Separately

This section describes the compilation and build of OAM components separately on Linux.

1. Execute the following command to clean the directories:

* **make clean PRODUCT=generic HARDWARE=generic BUILD=i686-linux**

1. Execute the following command to build individual components:

* **make PRODUCT=generic HARDWARE=generic BUILD=i686-linux**

1. Execute the following commands to build WDApp:

* **cd rsys/watchdog/**
* **make PRODUCT=generic HARDWARE=generic BUILD=i686-linux**

The WDApp binary ‘**watchdogd**’ is placed in the **rsys/watchdog/obj/generic/generic/i686-linux/** directory

1. Execute the following commands to build all the components:

* **cd rsys/tenb\_commonplatform/software/products/fap/hbs2/hbs2-iu/**
* **make distro PRODUCT=generic HARDWARE=generic BUILD=i686-linux**
* **cd rsys/tenb\_commonplatform/scripts/**

Execute the following command to fetch the generated libraries and binaries, and place in a common path.

* **TeNBCP\_linux\_load/TeNBCP\_arm\_load**

1. Execute the following commands to convert the scripts under the **/tenb\_oam** directory:

* **dos2unix scripts**

1. Execute the following script:

* **. setup\_TeNB**

**Note**: Use “**.**” and DO NOT use **./** or **sh** as the environment path is not set. This is script must be executed ONE time only (unless the test machine on which it is executing is changed).

For ARM compilation, execute the following command instead of the Linux command:

* **make PRODUCT=hbs2-iu HARDWARE=hbs2-4 BUILD=arm-none-linux-gnueabi**

# Executing OAM Components

The section describes the procedures to execute the OAM components.

## 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**

## 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**

# 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/stall.

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

Figure-2: Functional Architecture of Watchdog Application

**HeNB-App**

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.

## 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-11: WATCHDOG\_ADD\_PID\_REQ

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

1. 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-12: WATCHDOG\_REMOVE\_PID\_REQ

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

1. 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-13: WATCHDOG\_KICK\_REQ

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

1. 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-14: WATCHDOG\_KICK\_STOP\_REQ

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

## 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

## 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).

1. 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).

1. Path to save the log files using **wr\_cfg.txt** configuration file:
2. 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).

## 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/action based on the nature of the alarm.

# 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**

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

* **oam.set <parameter\_name> <value> - Set the parameter**

1. 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\_\***

1. 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.

1. 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-15: 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. |
| 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. |
| 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. |

## 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\_11CliArgumentESaIS2\_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

## 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:

1. ./libthreeway-system.so(Trace\_TraceAssert+0x64a) [0xb741113e]
2. ./libthreeway- messaging.so(\_ZN8threeway10CliHandler12CliCmdAssertERKSt6vectorINS\_11C liArgumentESaIS2\_EE+0x7e) [0xb770fad4]
3. ./libthreeway-messaging.so(\_ZN8threeway10CliHandler13ExecuteCliCmdERKSsS2\_+0x43e) [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

## 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:**

1. ./libthreeway-system.so(Trace\_TraceAssert+0x64a) [0xb735e13e]
2. ./libthreeway- messaging.so(\_ZN8threeway10CliHandler12CliCmdAssertERKSt6vectorINS\_11C liArgumentESaIS2\_EE+0x7e) [0xb765cad4]
3. ./libthreeway-messaging.so(\_ZN8threeway10CliHandler13ExecuteCliCmdERKSsS2\_+0x43e) [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

## 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:

1. ./libthreeway-system.so(Trace\_TraceAssert+0x64a) [0xb731313e]
2. ./libthreeway- messaging.so(\_ZN8threeway10CliHandler12CliCmdAssertERKSt6vectorINS\_11C liArgumentESaIS2\_EE+0x7e) [0xb7789ad4]
3. ./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

## 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**

1. 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.
2. 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**

# 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:**

1. 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.

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

[http://172.27.4.244/management](ftp://xmlsoft.org/libxml2/)/index.php

1. 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).

1. Enter the username and password in the **Managed Devices** tab in the ACSLite application:
2. 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*.*

# Appendix – B: Integration with Open Source Components

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

* 1. **MD5**
     + 1. **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);**

* + - 1. **md5/md5.h**

Include the following file:

**#include <stdint.h>**

Replace all **u\_char** with **uint8\_t** across the file.

* 1. **Kb\_getc**

1. **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;**

* 1. **libsoap**

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

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

**#ifdef \_\_cplusplus**

**}**

**#endif**

1. **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;**

**}**

1. **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;**

**}**

* 1. **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"**

* 1. **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;**

**}**

* 1. **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));**

**}**

* 1. **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);**

# Appendix – C: FAQs

## 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:

1. TeNB\_OAM\_Integration\_Guide\_1555464.pdf
2. TeNB\_OAM\_User\_Guide\_1222464.pdf
3. TeNB\_OAM\_Supported\_Parameters\_API\_Definition\_1100105.xlsx

## 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.

# 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



[**www.radisys.com**](http://www.radisys.com)