LTE TOTALeNodeB OAM

Integration Guide

1555464 5.0

Contents

Figures

Tables

1 Introduction

1.1 Purpose

The Integration Guide details the:

- Procedures to integrate generic Operations, Administration and Maintenance (OAM) module with Radisys LTE TOTALeNodeB (TeNB),
- Interfaces required for Radisys TeNB Stack Manager (SM) to integrate generic OAM module (TR-069 or other models) with TeNB stack and application,
- Configurations and APIs used for integrating generic OAM with TeNB SM.

This document does not intend to include the various OAM functionalities and how to implement them, but lists the important functionalities.

1.2 Target Audience

The Integration Guide is directed for developers integrating generic/third-party OAM module with TeNB stack and application to provide a complete end-to-end solution.

1.3 References

[1] LTE_TeNB_Software_Architecture.docx document.

1.4 Standards

[1] [TR-196], “Femto Access Point Service Data Model”, Broadband Forum, Issue 2, November 2011.

[2] [TR-069], “CPE WAN Management Protocol”, Broadband Forum, Issue 1, Amendment 4, July 2011.

1.5 Abbreviations and Acronyms

The abbreviations and acronyms used in this document are listed in Table-1.

Table-1: Acronyms

Acronym	Description
App	Sample Application Layer
CC	Call Control
CL	Convergence Layer
CLI	Command Line Interface
DL	Downlink
eNB / eNodeB	E-UTRAN Node B
E-UTRAN	Evolved UTRAN
FDD	Frequency Division Duplex
FM	Fault Management
FSM	Finite State Machine
HeMS	Home eNodeB Management System
IE	Information Element
IP	Internet Protocol
IPsec	IP Security
KPI	Key Performance Indicators
L-ARM / LARM	Lower ARM
LTE	Long Term Evolution
MAC	Medium Access Control Protocol
MIB	Master Information Block
MME	Mobile Management Entity
OAM	Operations, Administration and Maintenance
OAM&P	Operations, Administration, Maintenance, and Provisioning
ODMA	Opportunity-Driven Multiple Access

Acronym	Description
PDCP	Packet Data Convergence Protocol
PHY	Physical Layer
PM	Performance Management
RAC	Radio Admission Control
REM	Radio Environment Monitoring
RLC	Radio Link Control Protocol
RPC	Remote Procedure Call
RRC	Radio Resource Control Protocol
SCTP	Stream Control Transmission Protocol
S-GW / SGW	Serving Gateway
SIB	System Information Block
SM	Stack Manager
SSI	System Services Interface
TCP	Transmission Control Protocol
TCP / IP	Transmission Control Protocol / Internet Protocol
TeNB / TOTALeNB	TOTALeNodeB
TTI	Transmission Timing Interval
U-ARM / UARM	Upper ARM
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

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

1.6 Release History

Table-2 lists the history of changes in successive revisions to this document.

Table-2: Release History

Revision	Date	Description
5.0	March 17, 2016	LTE TOTALeNodeB Solution release, version GA 5.0
4.1	February 19, 2015	LTE TOTALeNodeB Solution release, version EA 4.0
4.0	October 21, 2014	LTE TOTALeNodeB Solution release, version EA 4.0 Beta
3.01	July 24, 2014	LTE TOTALeNodeB FDD Solution release, version GA 3.0 HotFix-1
3.0	July 14, 2014	LTE TOTALeNodeB FDD Solution release, version GA 3.0
2.0	March 21, 2014	LTE TOTALeNodeB FDD Solution release, version GA 2.0
1.2	July 10, 2013	LTE TOTALeNodeB Solution release, version 2.0 Alpha
1.1	April 29, 2013	LTE TOTALeNodeB Solution release, version 1.1
1.0	March 20, 2013	LTE TOTALeNodeB Solution release, version 1.0

2 Architecture Overview

2.1 Integrated Software Architecture

This section introduces the TeNB solution functional components and interfaces in **Figure-1** of the integrated architecture.

The solution includes Radisys TeNB core stack components and generic OAM components. Radisys TeNB core stack components include eNodeB Stack Manager, eNodeB application, eNodeB protocol stack, eNodeB Convergence Layer (CL), and eNodeB System Services Interface (SSI) components.

Generic OAM components include OAM CL interface plus generic OAM components. Here, generic OAM components are developed by customers to integrate proprietary or third-party OAM components.

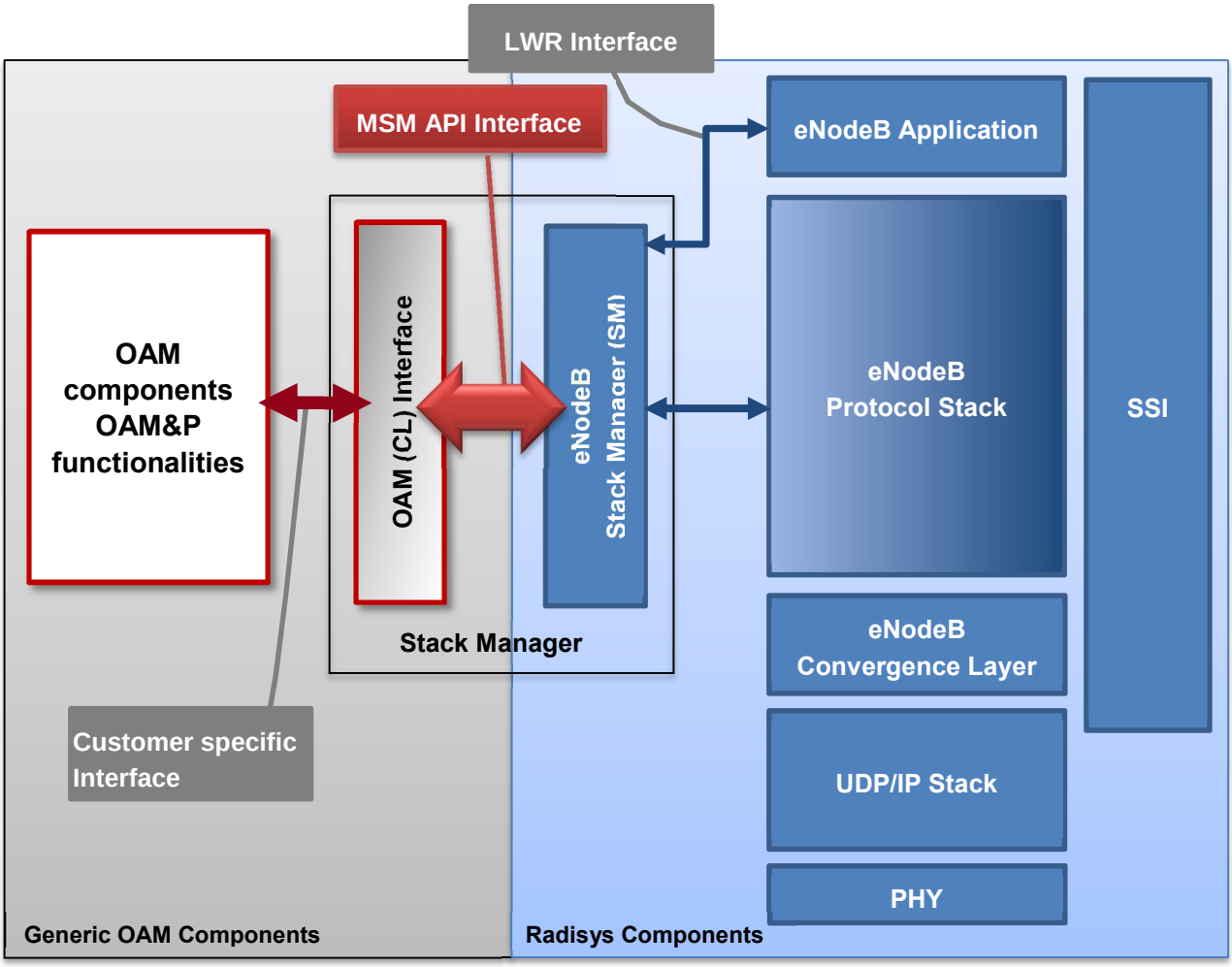
The OAM components provide:

- Configuration management services:
  - With interface towards small cell management system using TR-069 client module or proprietary interface
  - With procedures for initialization and custom deployment solutions.
- Performance management services with file upload and download functionalities
- Log management and tracing functionalities
- Security management of protected tunnel and authentication data
- Remote management with Command Line Interface (CLI) for debugging

The following modules must be implemented by customers integrating proprietary OAM components with TeNB stack and application:

- OAM CL interface
- Generic OAM components
- Remote Procedure Call (RPC) method between OAM CL interface and OAM components

Figure-1: Integrated Software Architecture



2.1.1 Radisys TeNB Components

The Radisys TeNB core stack components include:

- eNodeB application – Trillium Call Control (CC) applications and Finite State Machine (FSM)
- eNodeB protocol stack – 3GPP compliant Trillium eNodeB protocol stack
- eNodeB CL – Trillium silicon-specific eNodeB CL
- eNodeB SM – eNodeB SM is a Layer Manager (LM), which controls and interfaces with eNodeB application and eNodeB protocol stack layers. SM configures and manages all the protocol stack layers and eNodeB application.

  SM module implements the LM interface of all the protocol layers. All configuration requests are routed through the SM. The main responsibilities of the SM are:

  - Configure all the eNodeB layers
  - Control the Trillium protocol layers for binding
  - Handle alarms/fault events originating from eNodeB application and protocol stack

    - Handle events related to pegging of counters (part of the PM support functionality)
- PHY and third-party UDP/IP stack.

2.1.2 Generic OAM Components and OAM CL

OAM CL Interface

The CL abstracts the interface to the Stack Manager from the OAM components. The CL is designed as a library module integrating the generic OAM components with eNodeB SM using the MSM interface APIs. This interface is developed by the customer.

This method provides generic interface and flexibility to customers to develop inter module/process communication as it suites the generic OAM components design.

OAM Components

OAM components provide the OAM&P functionalities. OAM components are developed by customers to suite the OAM requirements. The functionalities include:

- Configuration Management
- Performance Management
- Fault Management

Along with further sub-functionalities like self-configuration, self-deployment procedures, and self-reconfiguration.

The architecture/design of these components is proprietary/customer specific.

OAM components must be able to configure and manage the eNodeB through the SM and interface with external OAM components like HeMS through TR-069 client module.

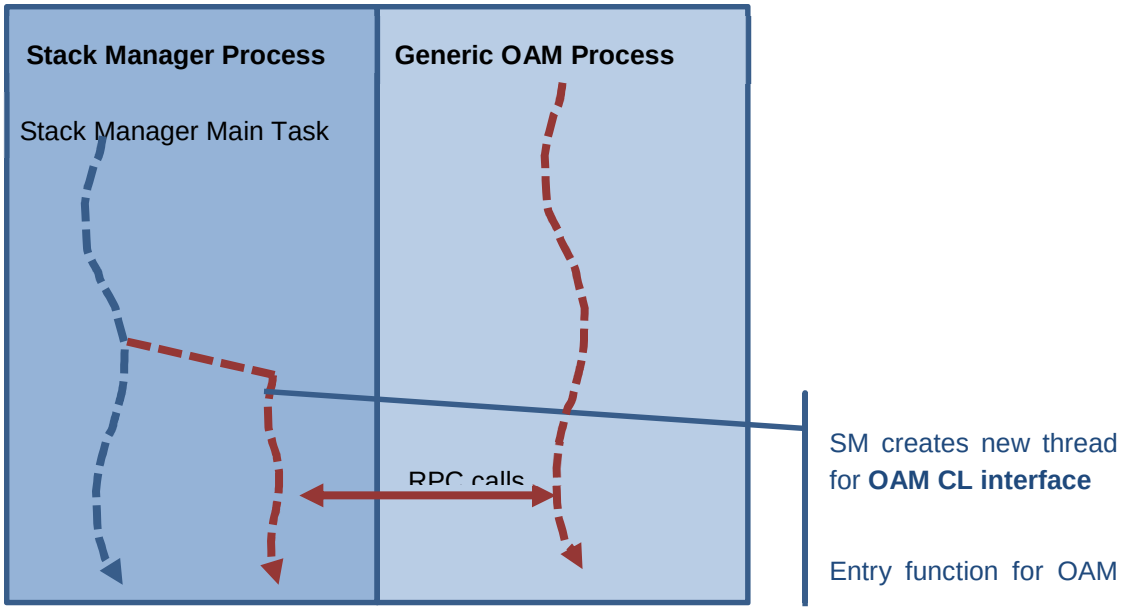
2.2 Stack Manager Thread Model

Stack Manager is responsible for initializing and creating the tasks for eNodeB stack and eNodeB application. SM creates a thread for OAM CL with the entry function as *int SmAppTst(void)*.

OAM CL interface must implement this entry function.

OAM CL thread is required for the OAM CL interface to implement RPC method to interact with OAM components. SM APIs for configuration towards eNodeB stack or eNodeB application must be invoked from this thread.

Figure-2: Stack Manager and OAM CL Thread Model



3 Stack Manager API

This section outlines the APIs provided by eNodeB SM to support OAM functionalities for configuration of eNodeB Application and eNodeB protocol stack and updating of performance counter values that are collected or measured.

3.1 Configuration Management

3.1.1 APIs for Initial eNodeB Configuration

The initial configuration parameters must be configured before the eNodeB cell is operational and start services. The configuration information is maintained in eNodeB SM (in lteeNodeBparams (base data type MsmLteeNodeBparams)). These data structures are accessed by Set and Get functions and OAM-CL cannot modify them. The whole set of configuration information must be provided to eNodeB SM from OAM-CL through set functions listed in Table-3. Individual set function must be called from OAM CL interface to propagate information from OAM-CL to SM for the respective group data.

After the initial configuration data is provided through set functions, the API MsmEnodeBinitialCfgComplete must be invoked to indicate to eNodeB SM that configuration parameters are initialized.

Any further configuration after eNodeB is up and running must be configured through MsmDynamicConfiguration API. The data stored in the lteeNodeBparams can be read by OAM-CL through the Get functions listed in Table-4.

Following APIs and data structures are provided to facilitate the parameter configuration.

The following APIs on the MSM interface must be used for setting all the initial configuration parameters. The allocation for the data structure passed in the Set functions must not be from heap and no free is called.

Table-3: Set APIs

Category	API Name	Description
Generic	S16 msmSetGenaricParameters(const MsmLteGenericParams *lteGenericParams)	To set generic parameters.
	S16 msmSetEnodebIpParameters(const MsmLteeNodeBIPparams *lteeNodeBIPparams)	To set MME IP and S1 connection mode.
	S16 msmSetAddCellConfigParameters(const MsmLteAddCellCfg *lteAddCellCfg)	To set parameters required to add a cell.
	S16 msmSetSmCellConfigParameters(const MsmLteSmCellCfg *lteSmCellCfg)	To set cell configuration parameters.
	S16 msmSetEnbProtoCfgParameters(const MsmLteEnbProtoCfg *lteEnbProtoCfg)	To set eNodeB protocol parameters.

Category	API Name	Description
	S16 msmSetFapControlParameters(const MsmLteFapControlParams *lteFapControlParams)	To set FAP control parameters.
	S16 msmSetMsCellCfgReqParameters(const MsmLteMsCellCfgReq *lteMsCellCfgReq)	To set MS cell configuration request parameters.
	S16 msmSetDbgParameters(const MsmLteDbgParams *lteDbgParams)	To set debug parameters.
	S16 msmSetUlAllocInfoCbParameters(const MsmLteUlAllocInfoCbParams *lteUlAllocInfoCbParams)	To set UL allocation information CB parameters.
	S16 msmSetProcTimerCfgParameters(const MsmLteProcTimerCfg *lteProcTimerCfg)	To set PROC timer configuration parameters.
	S16 msmSetEnbIpAddress(const MsmEnodebIpAddr *lteEnbIpAddr)	To set eNodeB IP address parameter.
	S16 msmSetPmParameters(const MsmPerfMgmtParameters *perfMgmtParameters)	To set performance management parameters.
	S16 msmSetFrequentFaultParameters(const MsmFrequentFaultMgmtParameters *frequentFaultMgmtParameters)	To set frequent fault parameters.
	S16 msmSetEarfcnRssiParameters(const MsmLteEarfcnRssiCfg *lteEarfcnRssiCfg);	To set EARFCN RSSI parameters.
MAC	S16 msmSetPrachConfigCommonParameters(const MsmLtePrachCfgCommon *ltePrachCfgCommon)	To set PRACH related parameters.
	S16 msmSetRachConfigCommonParameters(const MsmLteRachCfgCommonParams *lteRachCfgCommonParams)	To set RACH related parameters.
	S16 msmSetPdschConfigCommonParameters(const MsmLtePdschCfgCommon *ltePdschCfgCommon)	To set PDSCH common parameters.
	S16 msmSetUlPowerControlCommonParameters(const MsmLteULPwrCtrlCommon *lteULPwrCtrlCommon)	To set UL power control configuration parameters.
	S16 msmSetUlFrequencyInfoParameters(const MsmLteULFrequencyInfo *lteULFrequencyInfo)	To set UL frequency related parameters.
	S16 msmSetMacSchConfigParameters(const MsmLteMAC_SCH_Config *lteMAC_SCH_Config)	To set scheduler related parameters.
	S16 msmSetAntennaCommonConfigParameters(const MsmLteAntenna_Common_Config *lteAntenna_Common_Config)	To set Antenna configuration parameters.

Category	API Name	Description
	S16 msmSetPdschConfigDedicatedParameters(const MsmLtePdschCfgDedicated *ltePdschCfgDedicated)	To set PDSCH dedicated parameters.
	S16 msmSetPuschBasicCfgCommonParameters(const MsmLtePuschBasicCfgCommon *ltePuschBasicCfgCommon)	To set PUSCH common parameters.
	S16 msmSetPuschRefSignalCfgParameters(const MsmLtePuschRefSignalCfg *ltePuschRefSignalCfg)	To set PUSCH reference signal parameters.
	S16 msmSetPucchCfgCommonParameters(const MsmLtePucchCfgCommon *ltePucchCfgCommon)	To set PUCCH common parameters.
	S16 msmSetSrsCfgCommonParameters(const MsmLteSRSCfgCommon *lteSRSCfgCommon)	To set SRS configuration common parameters.
	S16 msmSetCqiPeriodicReportCfgParameters(const MsmLteCQIPeriodicReportCfg *lteCQIPeriodicReportCfg)	To set periodic CQI configuration related parameters.
	S16 msmSetPowerOffsetParameters(const MsmLtePowerOffsetParams *ltePowerOffsetParams)	To set power offset parameters.
	S16 msmSetPrbAllocationParameters(const MsmLtePRBallocation *ltePRBallocation)	To set PRB allocation parameters.
	S16 msmSetPcchConfigCommonParameters(const MsmLtePCCHCfgCommon *ltePCCHCfgCommon)	To set PCCH common configuration parameters.
	S16 msmSetCellUlSchdConfigParameters(const MsmLteCellUlSchdCfgGrp *lteCellUlSchdCfgGrp)	To set cell UL scheduler configuration parameters.
	S16 msmSetCellDlSchdConfigParameters(const MsmLteCellDlSchdCfgGrp *lteCellDlSchdCfgGrp)	To set cell DL scheduler configuration parameters.
	S16 msmSetSpsParameters(const MsmSpsCellConfigData *stSpsCellCfg)	To set SPS parameters.
RRC	S16 msmSetCellSib1ConfigGroupParameters(const MsmLteCellSib1CfgGrp *lteCellSib1CfgGrp)	To set SIB1 related parameters.
	S16 msmSetCell_Sib3ConfigGroupParameters(const MsmLteCellSib3CfgGrp *lteCellSib3CfgGrp)	To set SIB3 related parameters.
	S16 msmSetSib4CsgInfoParameters(const MsmLteSib4CsgInfo *lteSib4CsgInfo)	To set CSG information parameters.
	S16 msmSetSib5InterFreqCarrierInfoParameters(const MsmLteSib5InterFreqCarrierInfo *lteSib5InterFreqCarrierInfo)	To set SIB5 related Inter-frequency parameters.

Category	API Name	Description
	S16 msmSetCellSib6ConfigGroupParameters(const MsmLteCellSib6CfgGrp *lteCellSib6CfgGrp)	To set SIB6 related Inter UTRA frequency parameters.
	S16 msmSetSib8Parameters(const MsmCellSib8CfgGrp *cellSib8CfgGrp)	To set SIB8 parameters.
	S16 msmSetCellSib9CfgGrpParameters(const MsmLteCellSib9CfgGrp *lteCellSib9CfgGrp)	To set SIB9 related parameters.
	S16 msmSetDrxConfigParameters(const MsmLteDrxCfgParams *lteDrxCfgParams)	To set DRX related parameters.
	S16 msmSetCellMeasConfigParameters(const MsmLteCellMeasCfgGrp *lteCellMeasCfgGrp)	To set measurement related configuration parameters.
	S16 msmSetCellMibConfigParameters(const MsmLteCellMibCfgGrp *lteCellMibCfgGrp)	To set MIB parameters.
	S16 msmSetRabPolicyCfgGrpParameters(const MsmLteRabPolicyCfgGrp *lteRabPolicyCfgGrp)	To set the RAB policy group parameters.
	S16 msmSetUeTimerConstantsParameters(const MsmLteUeTimerConstants *lteUeTimerConstants)	To set UE timer constants.
	S16 msmSetCellEaidCfgGrpParameters(const MsmLteCellEAIDCfgGrp *lteCellEAIDCfgGrp)	To set EAID configuration parameters.
	S16 msmSetNeighFreqCfgParameters(const MsmLteNeighFreqCfg *lteNeighFreqCfg)	To set neighbor frequency related parameters.
	S16 msmSetNeighborListInUseParameters(const MsmNeighCellCfg *lteNeighCellCfg)	To set neighbor cell related parameters.
	S16 msmSetAcBarringInfoParameters(const MsmLteAcBarringInfo *lteAcBarringInfo)	To set access barring parameters.
	S16 msmSetSelfConfigParameters(const MsmLteSonGenericParams *lteSelfConfigParams)	To set self-configuration parameters.
	S16 msmSetCdmaBandClassParameters(const MsmCdmaBandClass *cdmaBandClass)	To set CDMA band class parameters.
	S16 msmSetCdmaMobilityParameters(const MsmCdmaMobilityParam *cdmaMobilityParam)	To set CDMA mobility parameters.
	S16 msmSetBarringCfg1xRtt(const MsmBarringConfig1xRTTCfg *barringConfig1xRTTCfg)	To set barring configuration for 1xRTTparameters.
	S16 msmSetCdma1XRttNghCell(const MsmNeighCellCfg *lteNeighCellCfg)	To set CDMA 1xRTT neighbor parameters.
	S16 msmSetCsfbCfgParameters(const MsmCsfbCfgGrp *csfbCfgGrp)	To set CSFB configuration parameters.

Category	API Name	Description
	S16 msmSetRimParameters(MsmLteRimParams* lteRimParams);	To set RIM parameters.
	S16 msmSetSchdPwrCfgParameters(MsmCellSchdPwrCfg *ltecellSchdPwrCfg)	To set scheduled power configuration parameters.
	S16 msmSetTddParameters(MsmLteTddParam *lteTddParam)	To set TDD parameters.
	S16 msmSetcsfbGeranCfgParameters(MsmLteCsfbGeranCfg *csfbToGeranCfg);	To set CSFB GERAN parameters.
	S16 msmSetNeighFreqCfgGeranParameters(const MsmLteNeighFreqCfg *lteNeighFreqCfg);	To set GERAN neighbor frequency parameters.
	S16 msmSetNeighCellCfgGeranParameters(const MsmNeighCellCfg *lteNeighCellCfg);	To set GERAN neighbor cell parameters.
	S16 msmSetTdsCdmaUtranTddFreqParameters(const MsmLteNeighFreqCfg *lteNeighFreqCfg)	To set TDS-CDMA neighbor frequency parameters.
	S16 msmSetTdsCdmaUtranTddCellParameters(const MsmNeighCellCfg *lteNeighCellCfg)	To set TDS-CDMA neighbor cell parameters.
REM	S16 msmSetRemControlParameters(const MsmLteRemControlParameters *lteRemControlParameters)	To set REM control parameters.
	S16 msmSetRemScanParameters(const MsmLteRemScanParameters *lteRemScanParameters)	To set REM scan parameters.
	S16 msmSetRemUtraControlParameters(const MsmRemUtraControlParameters *remUtraControlParameters)	To set REM UTRA control parameters.
	S16 msmSetRemScanUtraParameters(const MsmRemScanUtraParameters *remScanUtraParameters)	To set REM UTRA scan parameters.
	S16 msmSetFactoryRemParameters(const MsmLteRemControlParameters *lteRemControlParameters)	To set factory default parameters for REM
eGTP	S16 msmSetTunnelInfoParameters(const MsmLteTunnelInfoParams *lteTunnelInfoParams)	To set tunnel information parameters.
RRM	S16 msmSetRrmTddParameters(const MsmLteRrmTddParam *lteRrmTddParam)	To set RRM TDD parameters.

Category	API Name	Description
X2AP	S16 msmSetX2ConfigParameters(const MsmLteX2ConfigParams *lteX2ConfigParams)	To set X2 configuration parameters.
SCTP	S16 msmSetSctpConfigParameters(const MsmLteSctpCfgParams *lteSctpCfgParams)	To set SCTP configuration parameters.

Note: The data structure used for neighbor frequency (**MsmLteNeighEutraFreq**) contains serving frequency information at the 0th index and neighbor frequency information from the next index onwards. When a third party OAM updates this structure, it must ensure that the 0th index contains serving frequency information and the remaining indices contain neighbor frequency information. Application internally reads 0th index values and uses it for SIB3 broadcast and the rest for SIB5 broadcast. Currently, the data structures **MsmLteSib3ServFreqInfo** and **MsmLteSib3IntraFreqCellResel** for SIB3 configuration and **MsmLteSib5InterFreqCarrierInfo** for SIB5 configuration are not used, and are implemented for future purposes.

Table-4: Get APIs

Note: For all the following *get* functions, OAM-CL must allocate memory to receive the configuration data.

Category	API	Description
Generic	S16 msmGetGenaricParameters(MsmLteGenericParams *lteGenericParams)	To get generic parameters like Action type, Maximum TX power, BandList supported.
	S16 msmGetEnodebIpParameters(MsmLteeNodeBIPparams *lteeNodeBIPparams)	To get MME IP and S1 connection mode.
	S16 msmGetAddCellConfigParameters(MsmLteAddCellCfg *lteAddCellCfg)	To get parameters required to add a cell.
	S16 msmGetSmCellConfigParameters(MsmLteSmCellCfg *lteSmCellCfg)	To get cell configuration parameters.
	S16 msmGetEnbProtoCfgParameters(MsmLteEnbProtoCfg *lteEnbProtoCfg)	To get eNodeB protocol parameters.
	S16 msmGetFapControlParameters(MsmLteFapControlParams *lteFapControlParams)	To get FAP control parameters.
	S16 msmGetMsCellCfgReqParameters(MsmLteMsCellCfgReq *lteMsCellCfgReq)	To get MS cell configuration request parameters.
	S16 msmGetDbgParameters(MsmLteDbgParams *lteDbgParams)	To get debug parameters.

Category	API	Description
	S16 msmGetUlAllocInfoCbParameters(MsmLteUlAllocInfoCbParams *lteUlAllocInfoCbParams)	To get UL allocation information CB parameters.
	S16 msmGetProcTimerCfgParameters(MsmLteProcTimerCfg *lteProcTimerCfg)	To get PROC timer configuration parameters.
	S16 msmGetEnbIpAddress(MsmEnodebIpAddr *lteEnbIpAddr)	To get eNodeB IP address parameter.
	S16 msmGetFrequentFaultParameters(MsmFrequentFaultMgmtParameters *frequentFaultMgmtParameters)	To get frequent fault parameters.
	S16 msmGetEarfcnRssiParameters(const MsmLteEarfcnRssiCfg *lteEarfcnRssiCfg);	To get EARFCN RSSI parameters.
MAC	S16 msmGetPmParameters(MsmPerfMgmtParameters *perfMgmtParameters)	To get performance management parameters.
	S16 msmGetRachConfigCommonParameters(MsmLteRachCfgCommonParams *lteRachCfgCommonParams)	To get RACH related parameters.
	S16 msmGetPdschConfigCommonParameters(MsmLtePdschCfgCommon *ltePdschCfgCommon)	To get PDSCH common parameters.
	S16 msmGetUlPowerControlCommonParameters(MsmLteULPwrCtrlCommon *lteULPwrCtrlCommon)	To get UL power control configuration parameters.
	S16 msmGetUlFrequencyInfoParameters(MsmLteULFrequencyInfo * lteULFrequencyInfo)	To get UL frequency related parameters.
	S16 msmGetMacSchConfigParameters(MsmLteMAC_SCH_Config *lteMAC_SCH_Config)	To get scheduler related parameters.
	S16 msmGetAntennaCommonConfigParameters(MsmLteAntenna_Common_Config *lteAntenna_Common_Config)	To get Antenna configuration parameters.
	S16 msmGetPdschConfigDedicatedParameters(MsmLtePdschCfgDedicated *ltePdschCfgDedicated)	To get PDSCH dedicated parameters.
	S16 msmGetPuschBasicCfgCommonParameters(MsmLtePuschBasicCfgCommon *ltePuschBasicCfgCommon)	To get PUSCH common parameters.

Category	API	Description
	S16 msmGetPuschRefSignalCfgParameters(MsmLtePuschRefSignalCfg *ltePuschRefSignalCfg)	To get PUSCH reference signal parameters.
	S16 msmGetPucchCfgCommonParameters(MsmLtePucchCfgCommon *ltePucchCfgCommon)	To get PUCCH common parameters.
	S16 msmGetSrsCfgCommonParameters(MsmLteSRSCfgCommon *lteSRSCfgCommon)	To get SRS configuration common parameters.
	S16 msmGetCqiPeriodicReportCfgParameters(MsmLteCQIPeriodicReportCfg *lteCQIPeriodicReportCfg)	To get periodic CQI configuration related parameters.
	S16 msmGetPowerOffsetParameters(MsmLtePowerOffsetParams *ltePowerOffsetParams)	To get power offset parameters.
	S16 msmGetPrbAllocationParameters(MsmLtePRBallocation *ltePRBallocation)	To get PRB allocation parameters.
	S16 msmGetPcchConfigCommonParameters(MsmLtePCCHCfgCommon *ltePCCHCfgCommon)	To get PCCH common configuration parameters.
	S16 msmGetCellUlSchdConfigParameters(MsmLteCellUlSchdCfgGrp *lteCellUlSchdCfgGrp)	To get cell UL scheduler configuration parameters.
	S16 msmGetCellDlSchdConfigParameters(MsmLteCellDlSchdCfgGrp *lteCellDlSchdCfgGrp)	To get cell DL scheduler configuration parameters.
	S16 msmGetSpsParameters(MsmSpsCellConfigData * stSpsCellCfg)	To get SPS parameters.
RRC	S16 msmGetCellSib1ConfigGroupParameters(MsmLteCellSib1CfgGrp *lteCellSib1CfgGrp)	To get SIB1 related parameters.
	S16 msmGetCellSib3ConfigGroupParameters(MsmLteCellSib3CfgGrp *lteCellSib3CfgGrp)	To get SIB3 related parameters.
	S16 msmGetSib4CsgInfoParameters(MsmLteSib4CsgInfo *lteSib4CsgInfo)	To get CSG information parameters.

Category	API	Description
	S16 msmGetSib5InterFreqCarrierInfoParameters(MsmLteSib5InterFreqCarrierInfo *lteSib5InterFreqCarrierInfo)	To get SIB5 related Inter-frequency parameters.
	S16 msmGetCellSib6ConfigGroupParameters(MsmLteCellSib6CfgGrp *lteCellSib6CfgGrp)	To get SIB6 related Inter UTRA frequency parameters.
	S16 msmGetSib8Parameters(MsmCellSib8CfgGrp *cellSib8CfgGrp)	To get SIB8 parameters.
	S16 msmGetCellSib9CfgGrpParameters(MsmLteCellSib9CfgGrp *lteCellSib9CfgGrp)	To get SIB9 related parameters.
	S16 msmGetDrxConfigParameters(MsmLteDrxCfgParams * lteDrxCfgParams)	To get DRX related parameters.
	S16 msmGetCellMeasConfigParameters(MsmLteCellMeasCfgGrp *lteCellMeasCfgGrp)	To get measurement related configuration parameters.
	S16 msmGetCellMibConfigParameters(MsmLteCellMibCfgGrp *lteCellMibCfgGrp)	To get MIB parameters.
	S16 msmGetRabPolicyCfgGrpParameters(MsmLteRabPolicyCfgGrp *lteRabPolicyCfgGrp)	To get the RAB policy group parameters.
	S16 msmGetUeTimerConstantsParameters(MsmLteUeTimerConstants *lteUeTimerConstants)	To get UE timer constants.
	S16 msmGetCellEaidCfgGrpParameters(MsmLteCellEAIDCfgGrp *lteCellEAIDCfgGrp)	To get EAID configuration parameters.
	S16 msmGetNeighFreqCfgParameters(MsmLteNeighFreqCfg *lteNeighFreqCfg)	To get neighbor frequency related parameters.
	S16 msmGetNeighborListInUseParameters(MsmNeighCellCfg *lteNeighCellCfg)	To get neighbor cell related parameters.
	S16 msmGetAcBarringInfoParameters(MsmLteAcBarringInfo *lteAcBarringInfo)	To get access barring parameters.

Category	API	Description
	S16 msmGetSelfConfigParameters(MsmLteSonGenericParams *lteSelfConfigParams)	To get self-configuration parameters.
	S16 msmGetCdmaBandClassParameters(MsmCdmaBandClass * cdmaBandClass)	To get CDMA band class parameters.
	S16 msmGetCdmaMobilityParameters(MsmCdmaMobilityParam * cdmaMobilityParam)	To get CDMA mobility parameters.
	S16 msmGetBarringCfg1xRtt(MsmBarringConfig1xRTTCfg * barringConfig1xRTTCfg)	To get barring configuration for 1xRTTparameters.
	S16 msmGetCdma1XRttNghCell(MsmNeighCellCfg * lteNeighCellCfg)	To get CDMA 1xRTT neighbor parameters.
	S16 msmGetCsfbCfgParameters(MsmCsfbCfgGrp * csfbCfgGrp)	To get CSFB configuration parameters.
	S16 msmGetRimParameters(MsmLteRimParams* lteRimParams);	To get RIM parameters.
	S16 msmGetSchdPwrCfgParameters(MsmCellSchdPwrCfg *lteCellSchdPwrCfg)	To get scheduled power configuration parameters.
	S16 msmGetTddParameters(MsmLteTddParam *lteTddParam);	To get TDD parameters.
	S16 msmGetcsfbGeranCfgParameters(MsmLteCsfbGeranCfg *csfbToGeranCfg);	To get CSFB GERAN parameters.
	S16 msmGetNeighFreqCfgGeranParameters(MsmLteNeighFreqCfg *lteNeighFreqCfg);	To get GERAN neighbor frequency parameters.
	S16 msmGetNeighCellCfgGeranParameters(MsmNeighCellCfg *lteNeighCellCfg);	To get GERAN neighbor cell parameters.
	S16 msmGetTdsCdmaUtranTddFreqParameters(MsmLteNeighFreqCfg *lteNeighFreqCfg)	To get TDS-CDMA neighbor frequency parameters.
	S16 msmGetTdsCdmaUtranTddCellParameters(MsmNeighCellCfg *lteNeighCellCfg)	To get TDS-CDMA neighbor cell parameters.

Category	API	Description
REM	S16 msmGetRemControlParameters(MsmLteRemControlParameters *lteRemControlParameters)	To get REM control parameters.
	S16 msmGetRemScanParameters(MsmLteRemScanParameters *lteRemScanParameters)	To get REM scan parameters.
	S16 msmGetFactoryRemParameters(MsmLteRemControlParameters *lteRemControlParameters)	To get factory parameters.
	S16 msmGetRemUtraControlParameters(MsmRemUtraControlParameters *remUtraControlParameters)	To get REM UTRA control parameters.
	S16 msmGetRemScanUtraParameters(MsmRemScanUtraParameters *remScanUtraParameters)	To get REM UTRA scan parameters.
eGTP	S16 msmGetTunnelInfoParameters(MsmLteTunnelInfoParams *lteTunnelInfoParams)	To get tunnel information parameters.
RRM	S16 msmGetRrmTddParameters(MsmLteRrmTddParam *lteRrmTddParam)	To get RRM TDD parameters.
X2AP	S16 msmGetX2ConfigParameters(MsmLteX2ConfigParams *lteX2ConfigParams)	To get X2 configuration parameters.
SCTP	S16 msmGetSctpConfigParameters(MsmLteSctpCfgParams *lteSctpCfgParams)	To get SCTP configuration parameters.

Table-5: Initial eNodeB Configuration

API Details	Description
API Name	***MsmEnodeBinitialCfgComplete***
Synopsis	This Stack Manager API is a direct function called from OAM CL interface. This API is used to inform the eNodeB SM that configuration parameters are set using API defined in Table-3 and start configuration of eNB application and eNB stack.
Includes	wr_msm_common.h
Syntax	***S16 MsmEnodeBinitialCfgComplete(void)***

API Details	Description
Arguments	None.
Description	This API is used to indicate the eNodeB SM and eNodeB application to start configuration of each eNodeB protocol layer and proceed with cell setup. The **lteeNodeBparams** data structure must be configured using set API before calling the (base data type **MsmLteeNodeBparams**) API. This control API is used to synchronize eNB application with MIB. OAM component receives the configuration values from HeMS after initialization of eNodeB when the 'admin state' is set to 'unlocked' by HeMS.
Return Values	None.
Where to Use	This API is provided by SM and used by OAM CL interface.

Note: All the parameters in the **lteeNodeBparams** data structure must be populated before calling the **MsmEnodeBinitialCfgComplete** API. Default values are not assigned for these parameters.

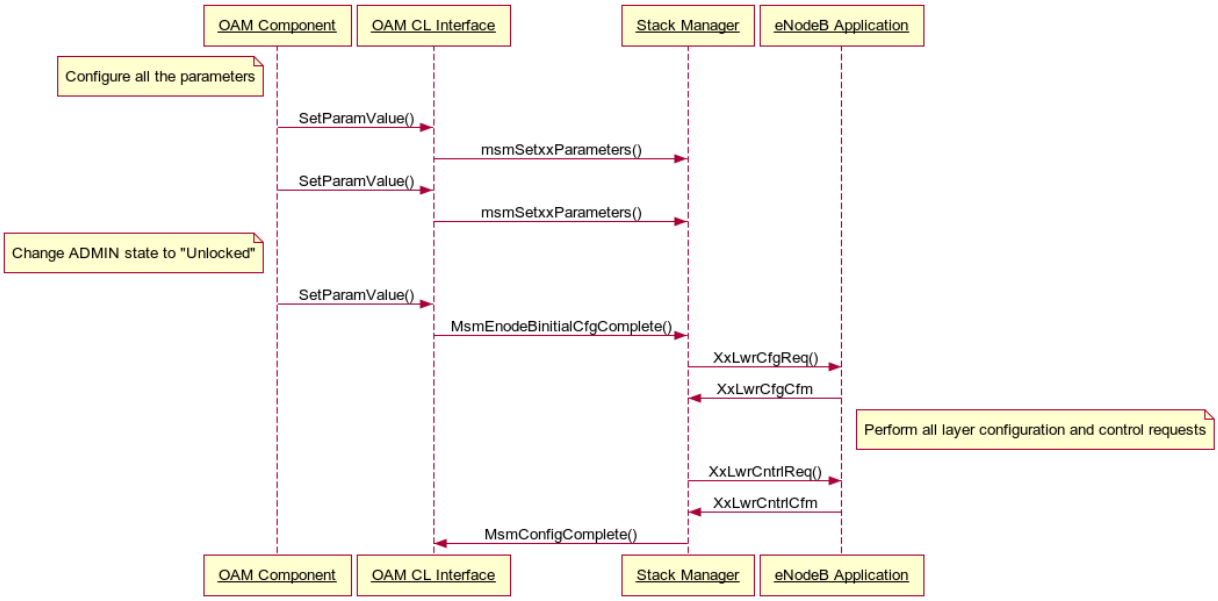
Table-6: eNodeB Configuration Complete

API Details	Description
API Name	***MsmConfigComplete***
Synopsis	This Stack Manager API is used to inform the OAM CL interface that the eNodeB application and eNodeB protocol layer configuration is completed.
Includes	wr_msm_common.h
Syntax	***void MsmConfigComplete(void)***
Arguments	None.
Description	Stack Manager completes configuring and initializing eNodeB application and eNodeB protocol layers, and indicates the OAM CL interface that configuration is complete. OAM CL interface must implement this function and wait for the indication before proceeding to further configuration, if required. Dynamic update is started after the initial configuration is completed.
Return Values	None.
Where to Use	This API is implemented by OAM CL interface and invoked by SM.

3.1.1.1 Sequence Diagram for Static Configuration

Figure-3 illustrates the call sequence of Static Configuration (Initial eNodeB Configuration and eNodeB Configuration Complete) APIs.

Figure-3: Static Configuration Call Sequence



Description:

1. Generic OAM component communicates configuration parameters to OAM CL interface when admin state is '**LOCK'**.
2. OAM CL interface configures these parameters to Stack Manager through the respective group of set function. For example: **msmSetGenericParameters()**
3. Generic OAM component communicates configuration parameters to OAM CL interface with admin state as '**UNLOCK'**.
4. OAM CL interface indicates configuration parameters are initialized to SM using **MsmEnodeBinitialCfgComplete()** function.
5. SM configures and initializes eNodeB application and eNodeB protocol layers.
6. SM calls **MsmConfigComplete()** function to indicate OAM Messenger that initial configuration is completed.

3.1.2 APIs for Dynamic Updating

Dynamic update helps to re-configure parameters after the eNodeB cell is operational with the following APIs and data structures.

API Details	Description
API Name	***MsmStartStopStackReq***
Synopsis	This SM API is a direct function called from OAM CL interface. This API is used to inform the eNodeB SM about the change in the admin state.

API Details	Description
Includes	msm_common.h
Syntax	***s16 MsmStartStopStackReq* (unsigned int isPremReq,unsigned int isStkSt,unsigned int isImd)**
Arguments	unsigned int isPremReq – To indicate whether stack state change is because of periodic REM scan. unsigned int isStkSt – To indicate start/stop stack(0 = stop, 1 = start) unsigned int isImd – To bring down stack immediately or after all UEs are released gracefully. (0 = deferred, 1 = immediate).
Description	This API is used to indicate the eNodeB stack manager to bring down/up after the eNodeB is up and operational.
Return Values	None.
Where To Use	This API is provided by SM and invoked by OAM CL interface.

API Details	Description
API Name	***MsmSmmDynCfgReq***
Synopsis	This SM API is a direct function called from OAM CL interface. This API is used to inform the eNodeB SM about the parameters that can be dynamically configured. For example: Dynamic MME list, neighbor cells, neighbor frequency, power parameters, SGW, and so on.
Includes	msm_common.h
Syntax	***MsmSmmDynCfgReq* (Void *cfg, U32 CfgType, bool Action)**
Arguments	**Void * cfg** – The configuration structure containing the updated values. This structure allocation must not be from heap and no free is called. ***Note**:* For dynamic configuration of Neighbor cell, the MSM structure *MsmDynNeighCellCfg* must be used. The static structure *MsmNeighCellCfg* must not be used. **U32 CfgType** – The configuration structure is of type enumeration defined by MsmSubscriptionGroup. For example: MSM_LTE_CELL_MIB_CONFIG_PARAMETERS is the SIB configuration structure. **bool Action** – This flag triggers the change to be applied immediately or deferred. This argument is currently not used.

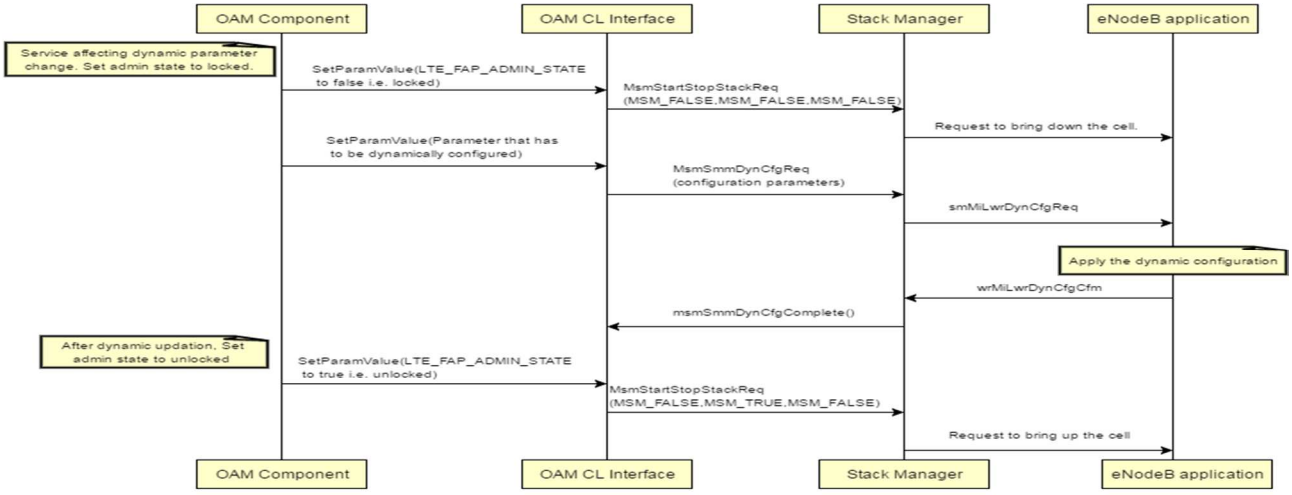
API Details	Description
Description	This SM API is invoked by OAM CL interface to inform eNodeB SM about the changes in the configuration parameters. This API accepts the pointer to structure and based on the CfgType, eNodeB SM and eNodeB application take required action.
Return Values	None.
Where To Use	This API is provided by SM and used by OAM CL interface.

API Details	Description
API Name	***msmDynConfigComplete***
Synopsis	This Stack Manager API is used to inform the OAM CL interface that the eNodeB application has completed the dynamic configuration and provides the status of success or failure
Includes	msm_common.h
Syntax	***msmDynConfigComplete*** **(U8 status, U8 cause, MsmDynCfmInfo cfmInfo)**
Arguments	**U8 status** – This is of type enumeration defined by MsmCfmType with values MSM_CFM_SUCCESS for success and MSM_CFM_FAILURE for failure. **U8 cause** – The cause value denotes the failure cause of type enumeration defined by MsmCauseType. **MsmDynCfmInfo cfmInfo** – This provides all the data related to the configuration like configuration type(as defined by MsmSubscriptionGroup), action (addition, modification, deletion) and key (used for multi-instance object)
Description	Stack Manager applies dynamic configuration at application and eNodeB protocol layers and indicates OAM CL interface that dynamic configuration is completed. OAM CL interface must implement this function and handle the status accordingly
Return Values	None.
Where To Use	This API is implemented by OAM CL interface and invoked by SM.

3.1.2.1 Sequence Diagram for Dynamic Configuration

Figure-4 illustrates the call sequence of Dynamic Configuration APIs.

Figure-4: Dynamic Updating Call Sequence



Description:

1. HeMS changes the admin state to **LOCK**, if dynamic update happens for any service affecting parameters.
2. OAM Messenger calls the ***MsmStartStopStackReq*** **()** function to request the eNodeB application to set the transmitter off.
3. HeMS configures the dynamic parameters and OAM Messenger calls the **MsmDynamicConfiguration()** function to send the changed parameters to eNodeB application.
4. Stack Manager responds to OAM CL interface on the status of the dynamic configuration through **msmDynConfigComplete()** function.
5. HeMS updates the admin state to **UNLOCK** when all the configuration parameters are populated.

Note: For parameters which are categorized as non-service effecting, ADMIN state LOCK and UNLOCK (Step 1 and step 3) are not required. The configuration can be applied dynamically.

3.2 Performance Management

This section describes the APIs used for Performance Management. eNodeB application and eNodeB protocol stack track Key Performance Indicators (KPIs). This is done by pegging each KPI in corresponding event. TeNB SM provides interface for incrementing respective KPI counter.

OAM component manages collection and processing of each of the KPIs.

3.2.1 API for Performance Counters

3.2.1.1 API for Performance Counters (L2/L3)

API Details	Description
API Name	***IncFapKpiByIntVal***
Synopsis	This API is a direct function call invoked by SM to update OAM CL about the KPI value change for a specific KPI.
Includes	wr_kpi.h
Syntax	***void IncFapKpiByIntVal (unsigned int KPI_ID, unsigned int incVal)***
Arguments	**KPI_ID** – This parameter indicates the KPI that is pegged. Each KPI is identified with a unique ID and defined in KpiId enumerator. **incVal** – This parameter indicates the integer value of the increment.
Description	This API is invoked by SM when eNodeB application or eNodeB stack pegging a particular KPI. This function must be implemented by OAM CL interface. When this API is invoked, the functionality must take care of processing the increment value for the specified KPI_ID. **Example**: SM invokes API as IncFapKpiByIntVal(KPI_ID_LTE_RRC_ATTCONESTAB_MTACCESS, 1).
Return Values	None.
Where to Use	This API is defined by OAM CL interface and TeNB SM invokes this API to increment the KPI counter. **Note**: This function call must be returned immediately as SM thread is blocked until this function returns.

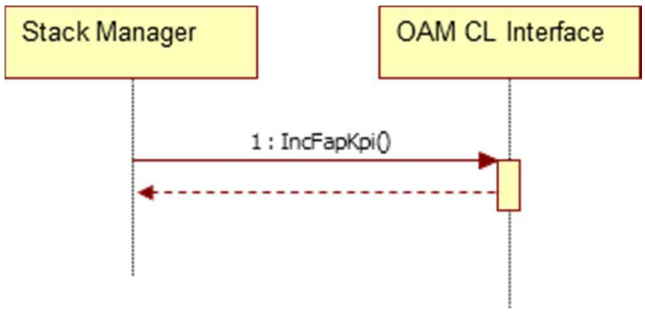
API Details	Description
API Name	***IncFapKpiByRealVal***
Synopsis	This API is a direct function call invoked by SM to update OAM CL about KPI value change for the specific KPI.
Includes	wr_kpi.h
Syntax	***IncFapKpiByRealVal (unsigned int KPI_ID, float incVal)***
Arguments	**KPI_ID** – This parameter indicates the KPI which is pegged. Each KPI is identified with a unique ID and defined in KPI_ID enumerator. **incVal** – This parameter indicates the real value increment.

API Details	Description
Description	This API is invoked by SM when eNodeB application or eNodeB stack pegging a particular KPI. This function must be implemented by OAM CL interface. When this API is invoked, the functionality must take care of processing the increment value for the specified KPI_ID.
Return Values	None.
Where to Use	This API is defined by OAM CL interface and TeNB SM invokes this API to increment the KPI counter. **Note**: This function call must be returned ASAP as SM thread is blocked until this function returns.

3.2.1.2 Sequence Diagram for KPI Pegging

Figure-5 illustrates the call sequence of KPI Pegging API.

Figure-5: KPI Pegging



Description:

1. eNodeB Stack Manager invokes **IncFapKpiByIntVal** or **IncFapKpiByRealVal** for each KPI_ID when a corresponding KPI event is noted at the eNodeB application or eNodeB stack.
2. OAM CL interface must handle this event by updating its data structures (done internally by generic OAM implementation).

3.2.2 API for L2 Counters

This section describes the APIs required for L2 Counters measurement and the usage:

- Start L2 Measurement Request: To inform the layers to initiate the L2 measurement request.
- Collect L2 Measurement Counters: To inform the eNB stack layers to provide the statistics for the defined collection period.
- Stop L2 Measurement Request: To inform the layers to stop the L2 Counter KPI measurement.

Table-7: Start L2 Measurement Request

API Details	Description
API Name	***wrKpiStartKpiCollecPrc***
Synopsis	This API is a direct function call invoked from OAM CL interface to inform the eNodeB stack layers through eNodeB SM to start the L2 measurement procedure.
Includes	wr_msm_common.h
Syntax	***S16 wrKpiStartKpiCollecPrc(bool)***
Arguments	**bool** – Flag set to 'TRUE' to start L2 measurement.
Description	This API is used to indicate the eNodeB stack layers (PDCP, RLC, and MAC) through eNodeB SM and eNodeB application to start the L2 measurement procedure. After receiving the Start L2 Measurement Request message, the layers start the L2 measurement procedures.
Return Values	None.
Where to Use	This API is provided by SM and used by OAM CL interface.

Table-8: Collect L2 Measurement Statistics

API Details	Description
API Name	***wrSendKpisInfo***
Synopsis	This API is a direct function call invoked from OAM CL interface to inform the eNodeB stack layers through eNodeB SM to send the L2 measurements.
Includes	wr_msm_common.h
Syntax	***S16 wrSendKpisInfo()***
Arguments	None.
Description	This API is used to indicate the eNodeB stack layers (PDCP, RLC, and MAC) through eNodeB SM and eNodeB application to collect the L2 measurement counters. OAM must call this API on expiry of collection period timer.
Return Values	None.
Where to Use	This API is provided by SM and used by OAM CL interface.

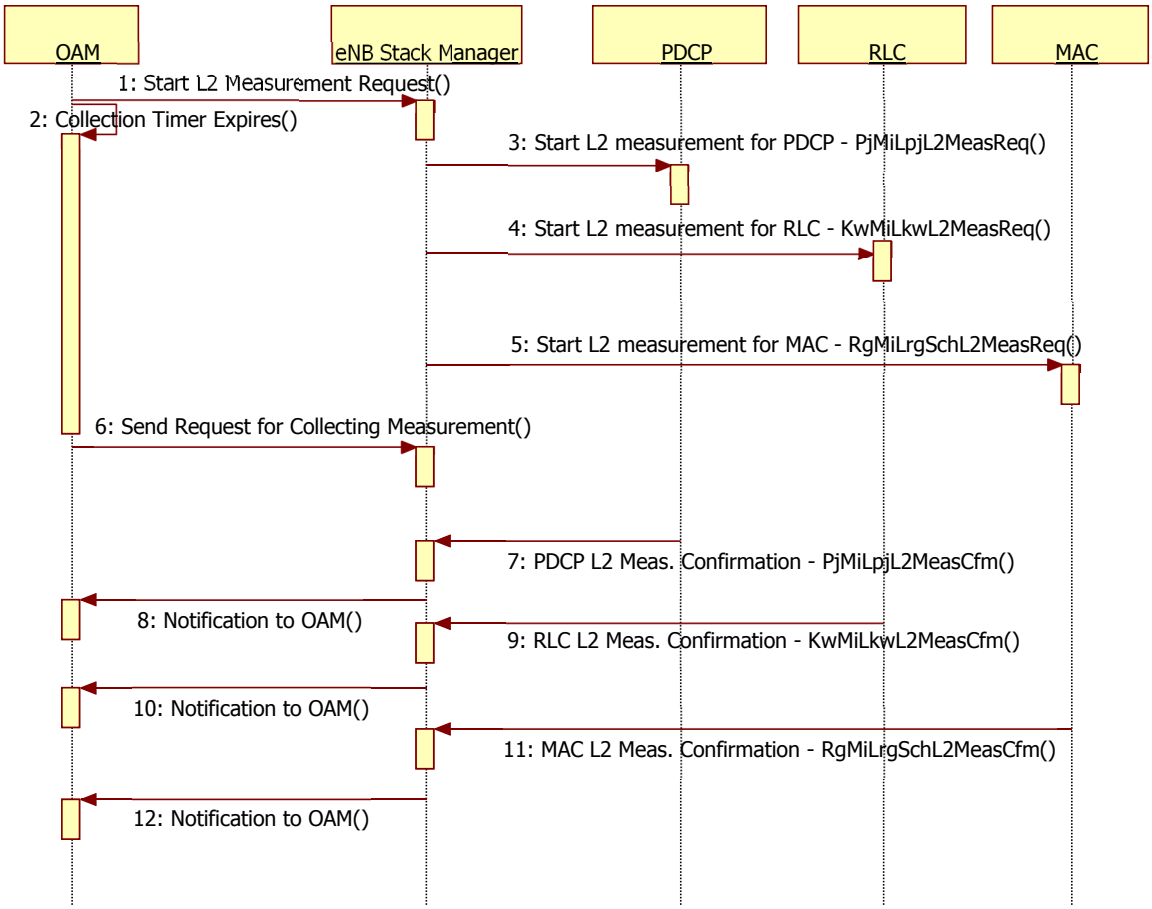
Table-9: Stop the L2 Measurement

API Details	Description
API Name	***wrKpiStopKpiCollecPrc***
Synopsis	This API is a direct function call invoked from OAM CL interface to inform the eNodeB stack layers through eNodeB SM to stop the L2 measurement procedure.
Includes	wr_msm_common.h
Syntax	***S16 wrKpiStopKpiCollecPrc()***
Arguments	None.
Description	This API is used to indicate the eNodeB stack layers (PDCP, RLC, and MAC) through eNodeB SM and eNodeB application to stop the L2 measurement procedure. Before sending the Stop L2 Measurement Request message, the OAM must send Collect Measurement Request message to the eNodeB stack layers.
Return Values	None.
Where to Use	This API is provided by SM and used by OAM CL interface.

3.2.2.1 Sequence Diagram for L2 Measurement Procedure

Figure-6 illustrates the call sequence of L2 Measurement APIs.

Figure-6: L2 Measurement Procedure



Description:

The steps for collecting the L2 measurement statistics are:

1. OAM sends the Start L2 Measurement Request message.
2. The request is forwarded to the respective layers, for example, RLC, PDCP, and MAC through Stack Manager.
3. When the collection timer expires, the OAM sends notification to SM for collecting L2 Counters Measurement.
4. Lower Layers (PDCP/RLC/MAC) send L2 Counters Measurements to SM.
5. SM updates L2 Counters Measurements in KPI collector as received from the lower layers.

3.3 Fault Management

This section describes the API used for Fault Management (FM).

eNodeB application and eNodeB protocol stack support FM alarms – SCTP link failure, X2 setup failure, and so on. This function is completed by alarm object definition, storing and reporting mechanisms as per TR-157 specification.

3.3.1 API for Fault Management

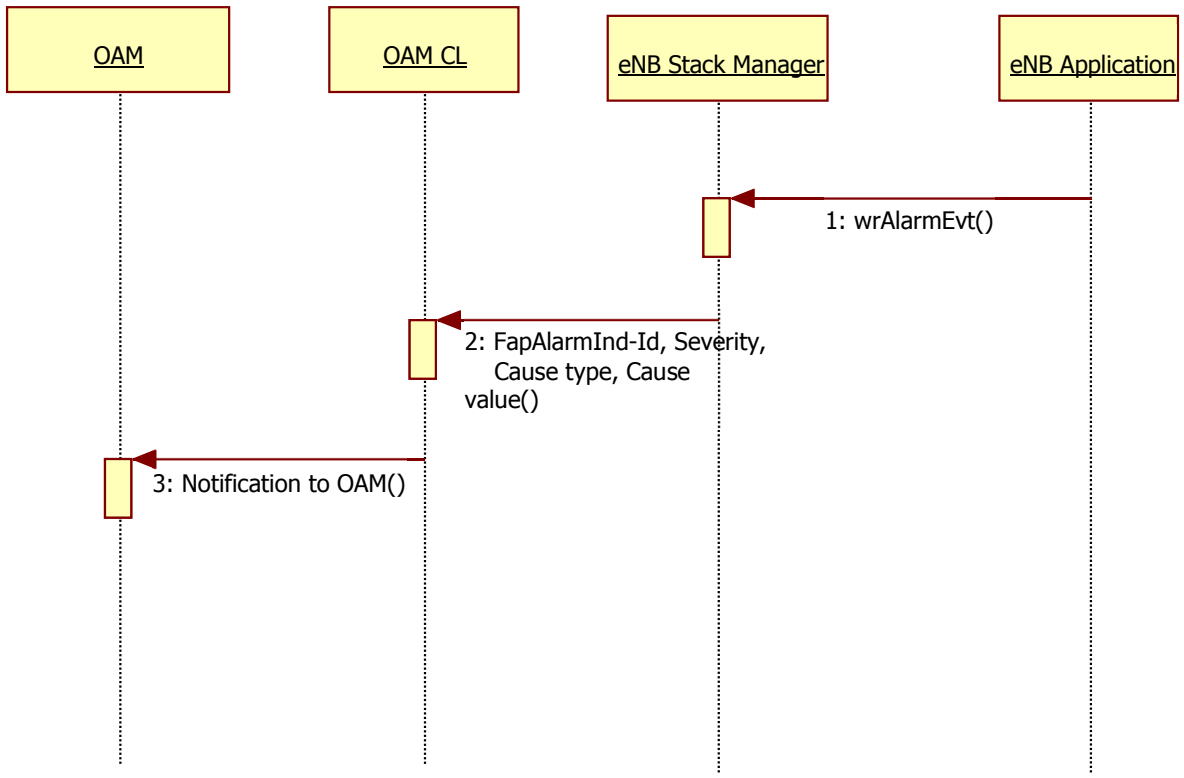
This section describes the API to be used for Fault Management.

API Details	Description
API Name	***FapAlarmInd***
Synopsis	This API is a direct function call invoked by SM to update OAM CL about the faults generated.
Includes	SmApplication.cpp
Syntax	***Void FapAlarmInd (u32 alrmaid, u8 severity, u8 cause type, u8 cause value)***
Arguments	**AlarmId** – ID of the alarm generated. **Severity** – Severity can be cleared, major, minor, or critical. **Cause type** – Cause of the fault generated. **Cause value** – Value of the cause.
Description	This API is used to report the faults generated from eNodeB application to OAM. This function must be implemented by OAM CL interface. When this API is invoked, the OAM Fault Management functionality takes care of processing the faults depending on the nature of the fault generated.
Return Values	None.
Where to Use	This API is provided by SM and used by OAM CL interface.

3.3.1.1 Sequence Diagram for Reporting of Fault

Figure-7 illustrates the call sequence of Fault Reporting APIs.

Figure-7: Reporting of Fault Procedure



Description:

The steps for reporting and collecting the fault are:

1. If the fault is generated at eNodeB application, eNodeB application calls wrAlarmEvt() function to pack the generated alarm in the status indication and forwards to eNodeB SM.
2. eNodeB SM unpacks the status indication and calls the alarm indication, fapAlarmInd() function. In fapAlarmInd() function, alarm ID, severity and the specific cause for the generated alarm is passed to OAM CL.

   For example: Alarm ID: 11110

   Severity: Major (3)

   Specific Cause: X2 setup failure

4 Integrating OAM CL with eNodeB

Following generic steps can be used to integrate OAM CL Interface with HeNB binary.

1. OAM CL to be built as a library (with makefiles, as required)
2. OAM CL library path to be included in the eNodeB build files.
3. Build eNodeB binary as mentioned in the Section 9 of *TeNB_OAM_User_Guide_1222464.pdf* document.

5 Features Supported

Table-10 lists the supported OAM features.

Table-10: Supported Features

Feature	Support	Remarks
Static configuration	Yes	Parameters are described in the *OAM_Compliance.xlsx* document.
Parameter validation	Yes	Parameters are described in the *OAM_Compliance.xlsx* document.
First time boot-up (BOOTSTRAP) procedure	Yes	Parameters for the first time initialization of the OAM.
BOOT procedure	Partial	Supports rebooting procedure of OAM and U-ARM. MSPD support is required for L-ARM rebooting.
Dynamic updating	Yes	1. MME addition/modification/deletion 2. Neighbour Cell 3. Neighbour eNB 4. Neighbour Frequency 5. Measurement related parameters (Idle mode).
TR-069 procedures	Yes	1. Active/Passive notification 2. Add/Delete object 3. Six Connection Requests
Command Line Interface (CLI)	Yes	Parameters are described in the *OAM_Compliance.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 *OAM_Compliance.xlsx* document for more details in the release package.

www.radisys.com