|  |
| --- |
|  |
| rsys_logo |
| **LTE TotaleNodeB Solution**  **Operations, Administration and Maintenance (OAM)**  **Integration Guide**  **1555464 2.0** |
|  |
|  |
|  |

© 2014 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 Introduction 5](#_Toc380148218)

[1.1 Purpose 5](#_Toc380148219)

[1.2 Target Audience 5](#_Toc380148220)

[1.3 References 5](#_Toc380148221)

[1.4 Standards 5](#_Toc380148222)

[1.5 Abbreviations and Acronyms 5](#_Toc380148223)

[1.6 Release History 9](#_Toc380148224)

[2 HeNB Architecture Overview 10](#_Toc380148225)

[2.1 Integrated Software Architecture 10](#_Toc380148226)

[2.1.1 Radisys TeNB Components 11](#_Toc380148227)

[2.1.2 Generic OAM Components and OAM CL 12](#_Toc380148228)

[2.2 Stack Manager Thread Model 13](#_Toc380148229)

[3 Stack Manager API 14](#_Toc380148230)

[3.1 Configuration Management 14](#_Toc380148231)

[3.1.1 APIs for Initial eNodeB Configuration 14](#_Toc380148232)

[3.1.1.1 Sequence Diagram for Static Configuration 15](#_Toc380148233)

[3.1.2 APIs for Dynamic Updating 16](#_Toc380148234)

[3.1.2.1 Sequence Diagram for Dynamic Configuration 18](#_Toc380148235)

[3.2 Performance Management 19](#_Toc380148236)

[3.2.1 API for Performance Counters 19](#_Toc380148237)

[3.2.1.1 API for Performance Counters (L2/L3) 19](#_Toc380148238)

[3.2.1.2 Sequence Diagram for KPI Pegging 20](#_Toc380148239)

[3.2.2 API for L2 Counters 21](#_Toc380148240)

[3.2.2.1 Sequence Diagram for L2 Measurement Procedure 22](#_Toc380148241)

[3.3 Fault Management 23](#_Toc380148242)

[3.3.1 API for Fault Management 23](#_Toc380148243)

[3.3.1.1 Sequence Diagram for Reporting of Fault 24](#_Toc380148244)

[4 Integrating OAM CL with eNodeB 25](#_Toc380148245)

[5 Features Supported 26](#_Toc380148246)

**Figures**

[Figure-1: Integrated Software Architecture 11](#_Toc381033128)

[Figure-2: Stack Manager and OAM CL Thread Model 13](#_Toc381033129)

[Figure-3: Static Configuration Call Sequence 15](#_Toc381033130)

[Figure-4: Dynamic Updating Call Sequence 18](#_Toc381033131)

[Figure-5: KPI Pegging 20](#_Toc381033132)

[Figure-6: L2 Measurement Procedure 22](#_Toc381033133)

[Figure-7: Reporting of Fault Procedure 24](#_Toc381033134)

**Tables**

[Table-1: Abbreviations and Acronyms 5](#_Toc381033266)

[Table-2: Release History 9](#_Toc381033267)

[Table-3: Initial eNodeB Configuration 14](#_Toc381033268)

[Table-4: eNodeB Configuration Complete 15](#_Toc381033269)

[Table-5: Start L2 Measurement Request 21](#_Toc381033270)

[Table-6: Collect L2 Measurement Statistics 21](#_Toc381033271)

[Table-7: Stop the L2 Measurement 22](#_Toc381033272)

# Introduction

## Purpose

The Integration Guide (IG) details the:

* procedures to integrate generic Operation, 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 ones.

## Target Audience

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

## References

[1] TeNB\_Software\_Architecture.pdf document.

## 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.

## Abbreviations and Acronyms

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

Table-1: Abbreviations and Acronyms

|  |  |
| --- | --- |
| **Acronym** | **Description** |
| ANR | Automatic Neighbor Relations |
| APN | Access Point Network |
| App | Sample Application Layer |
| ATCA | Advanced Telecom Computing Architecture |
| BCCH | Broadcast Control CHannel |
| BSP | Bi-criterion Shortest Path |
| CC | Call Control |
| CL | Convergence Layer |
| CLI | Command Line Interface |
| CM | Configuration Mode |
| CMAS | Commercial Mobile Alert System |
| CN | Core Network |
| CNE | Core Network Emulator |
| CPU | Central Processing Unit |
| CQI | Channel Quality Indicator |
| CWMP | Common WAN Management Protocol |
| DL | Downlink |
| DLPC | Downlink Power Control |
| DRB | Data Radio Bearer |
| DRX | Discontinuous Reception |
| DTCH | Dedicated Traffic CHannel |
| ECGI | E-UTRAN Cell Global Identifier |
| e-GTP | Evolved GTP |
| EARFCN | E-UTRA Absolute Radio Frequency Channel Number |
| eNB/eNodeB | E-UTRAN Node B |
| EPC | Evolved Packet Core |
| EPS | Evolved Packet System |
| E-RAB / ERAB | E-UTRAN Radio Access Bearer |
| ETWS | Earthquake and Tsunami Warning System |
| E-UTRAN | Evolved UTRAN |
| EVM | Error Vector Magnitude |
| FDD | Frequency Division Duplex |
| FDX | Full Duplex |
| FGI | Feature Group Indicator |
| FM | Fault Management |
| FSM | Finite State Machine |
| GTP | GPRS Tunneling Protocol |
| HARQ | Hybrid ARQ |
| HeMS | Home eNodeB Management System |
| HSS | Home Subscriber Server |
| IE | Information Element |
| IG | Integration Guide |
| IKEv2 | Internet Key Exchange v2 |
| IMSI | International Mobile Subscriber Identity |
| IP | Internet Protocol |
| IPsec | IP Security |
| KPI | Key Performance Indicators |
| LA | Link Adaptation |
| L-ARM / LARM | Lower ARM |
| LCG | Logical Channel Group |
| 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 |
| MTU | Maximum Transmission Unit |
| NAS | Non-access Stratum |
| NMM | Network Monitor Mode |
| NR | Neighbor Relation |
| NRT | Neighbor Relation Table |
| OAM | Operation, Administration and Maintenance |
| OAM&P | Operations, Administration, Maintenance, and Provisioning |
| ODMA | Opportunity-Driven Multiple Access |
| OPC | Operations Controller |
| PCI | Physical Cell Identifier |
| PDCCH | Physical Downlink Control CHannel |
| PDCP | Packet Data Convergence Protocol |
| PDN | Packet Data Network |
| PDSCH | Physical Downlink Shared CHannel |
| PDU | Protocol Data Unit |
| PFS | Proportional Fair Scheduler |
| PHY | Physical Layer |
| PM | Performance Management |
| PUSCH | Physical Uplink Shared CHannel |
| P-GW/PDN-GW/PGW | PDN Gateway |
| PLMN | Public Land Mobile Network |
| PWS | Public Warning System |
| QPSK | Quadrature Phase Shift Keying |
| QXDM | Qualcomm Extensible Diagnostic Monitor |
| RAB | Radio Access Bearer |
| 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 |
| SDU | Service Data Unit |
| SIB | System Information Block |
| SINR | Signal to Noise Ratio |
| SM | Stack Manager |
| SoC | System-on-a-Chip |
| SSI | System Services Interface |
| TCP | Transmission Control Protocol |
| TCP/IP | Transmission Control Protocol / Internet Protocol |
| TDD | Time Division Duplex |
| TeNB/TotaleNB | TotaleNodeB |
| TFTP | Trivial File Transfer Protocol |
| 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 |
| USB | Universal Serial Bus |
| UTRAN | Universal Terrestrial Radio Access Network |
| VC | Video Client |
| VS | Video Server |
| X2AP | X2 Application Protocol |

For a list of commonly used terms, refer to the Engineering Glossary at [www.radisys.com/resources/wireless-glossary/](http://www.radisys.com/resources/wireless-glossary/)

## Release History

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

Table-2: Release History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author (s) | Description |
| 2.0 | March 21, 2014 | Priya P. | Conforms to TotaleNodeB FDD Solution release, version GA 2.0 |
| 1.2 | July 10, 2013 | OAM Team | Conforms to TotaleNodeB Solution release, version 2.0 Alpha |
| 1.1 | April 29, 2013 | OAM Team | Conforms to TotaleNodeB Solution release, version 1.1 |
| 1.0 | March 20, 2013 | OAM Team | Initial draft. Conforms to TotaleNodeB Solution release, version 1.0 |

# HeNB Architecture Overview

## Integrated Software Architecture

This section introduces the TotaleNodeB 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/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 TotaleNodeB stack and application.

* OAM CL interface,
  + Generic OAM components,
  + Remote Procedure Call (RPC) method between OAM CL interface and OAM components.

**OAM components**

**OAM&P functionalities**

**SSI**

**Generic OAM Components**

**eNodeB Application**

**eNodeB**

**Protocol Stack**

**eNodeB Convergence Layer**

**UDP/IP Stack**

**PHY**

**Radisys Components**

**LWR Interface**

**Customer specific Interface**

**OAM (CL) Interface**

**eNodeB**

**Stack Manager (SM)**

**Stack Manager**

**MSM API Interface**

Figure-1: Integrated Software Architecture

### 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 is responsible for configuring and managing 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:

* Configuring all the eNodeB layers,
* Controlling the Trillium protocol layers for binding,
* Handling alarms/fault events originating from eNodeB application and protocol stack,
* Handling events related to pegging of counters (part of the PM support functionality).
  + PHY and third-party UDP/IP stack.

**Note**: For more information on each of the components, refer TeNB\_Software\_Architecture.pdf document.

### 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 to their 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, and
  + Fault Management 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.

## 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.

Stack Manager Main Task

SM creates new thread for **OAM CL interface**

Entry function for OAM CL interface thread is

int SmAppTst(void)

**Stack Manager Process**

**Generic OAM Process**

RPC calls

Figure-2: Stack Manager and OAM CL Thread Model

# 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.

## Configuration Management

### APIs for Initial eNodeB Configuration

The initial configuration parameters must be configured before the eNodeB cell is operational and start services.

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

Table-3: 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 initialized and start configuration of eNB application and eNB stack. |
| **Includes** | wr\_msm\_common.h |
| **Syntax** | ***S16 MsmEnodeBinitialCfgComplete(void)*** |
| **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 populated before calling the (base data type **MsmLteeNodeBparams**) API. This data structure is shared between SM task and OAM CL task, and 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. |

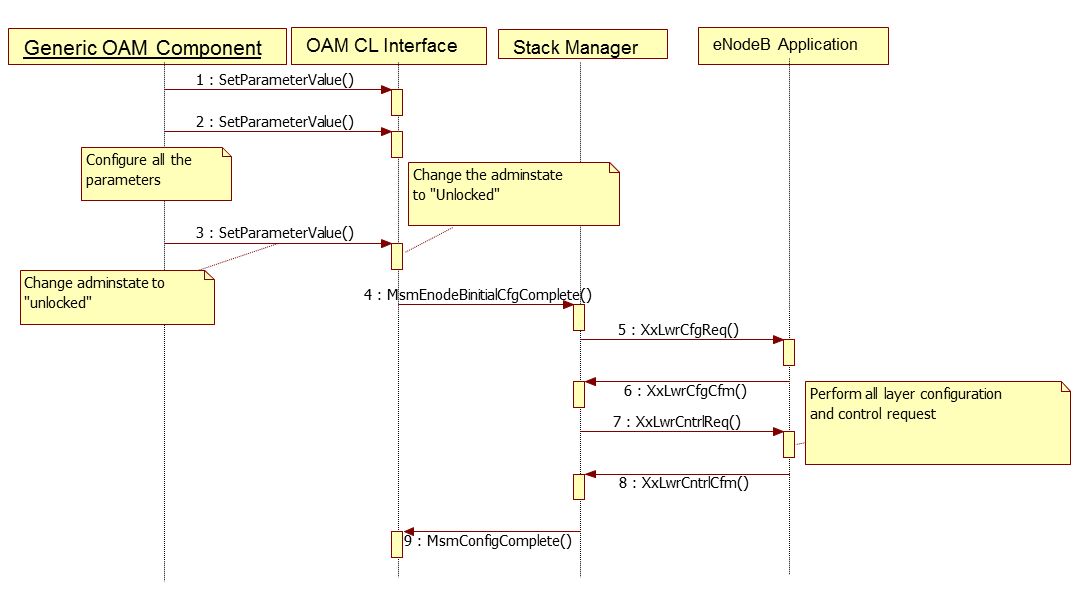
**Note**: All the parameters in the **lteeNodeBparams** data structure must be populated before calling the **MsmEnodeBinitialCfgComplete** API. Default values are not assigned for these parameters.

Table-4: 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 functionandwait for the indication before proceeding to further configuration, if required.  Dynamic updating 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. |

#### 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**

As an example, scenario in:

1. Generic OAM component communicates configuration parameters to OAM CL interface when admin state is ‘locked’.
2. Generic OAM component communicates configuration parameters to OAM CL interface with admin state as ‘unlocked’.
3. OAM CL interface indicates configuration parameters are initialized to SM using **MsmEnodeBinitialCfgComplete()**.
4. SM configures and initializes eNodeB application and eNodeB protocol layers.
5. SM calls **MsmConfigComplete()** to indicate OAM Messenger that initial configuration is completed.

### APIs for Dynamic Updating

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

|  |  |
| --- | --- |
| **API Details** | **Description** |
| **API Name** | ***MsmAdminStateChanged*** |
| **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. |
| **Includes** | msm\_common.h |
| **Syntax** | ***s16 MsmAdminStateChanged*** (unsigned char adminState) |
| **Arguments** | **unsigned char adminState** – The state of the admin toggles between locked and unlocked (0 = lock, 1 - unlock). |
| **Description** | This API is used to indicate the eNodeB SM that the admin state is changed for dynamic updating of parameters. If there is a change in service affecting parameter, the API is used to lock the admin state and later unlock after the changes are done. This prepares SM to allow further re-configuration of parameters.  Setting the adminState to ‘lock’ triggers the operational state as ‘down’ (that is, set radio Tx OFF).  Setting the adminState to ‘unlock’ triggers the operational state as ‘up’ (that is, set radio Tx ON). |
| **Return Values** | None. |
| **Where To Use** | This API is provided by SM and invoked by OAM CL interface. |

|  |  |
| --- | --- |
| **API Details** | **Description** |
| **API Name** | ***MsmDynamicConfiguration*** |
| **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** | ***MsmDynamicConfiguration*** (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.  **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. |
| **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. |

#### Sequence Diagram for Dynamic Configuration

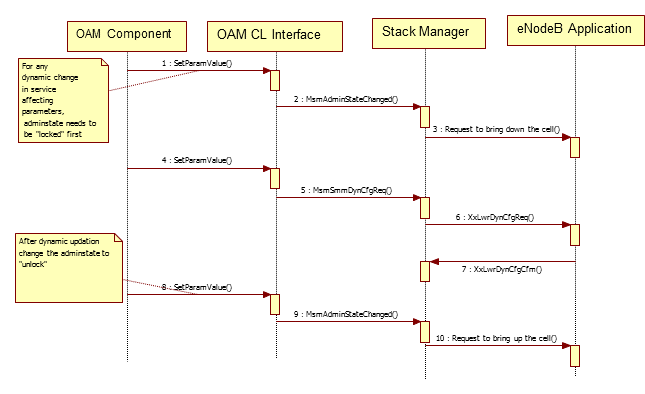


Figure-4: Dynamic Updating Call Sequence

Explanation:

1. If dynamic updating happens for any service affecting parameters, HeMS changes the admin state to **lock**.
2. OAM Messenger calls the **MsmAdminStateChanged()** to request the eNodeB application to set the transmitter off.
3. HeMS configures the dynamic parameters and OAM Messenger calls the **MsmDynamicConfiguration()**to send the changed parameters to eNodeB application.
4. HeMS updates the admin state to **unlock** when all the configuration parameters are populated.

## 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.

### API for Performance Counters

#### 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 TotaleNodeB 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 KpiId enumerator.  **incVal** – This parameter indicates the real value 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. |
| **Return Values** | None. |
| **Where to Use** | This API is defined by OAM CL interface and TotaleNodeB 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. |

#### Sequence Diagram for KPI Pegging

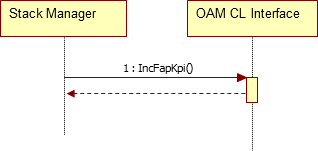


Figure-5: KPI Pegging

Explanation for Sequence 1:

1. eNodeB Stack Manager invokes either **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).

### API for L2 Counters

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

* 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-5: 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.  On 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-6: 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-7: 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. |

#### Sequence Diagram for L2 Measurement Procedure

OAM

eNB Stack Manager

PDCP

RLC

MAC

1: Start L2 Measurement Request()

2: Collection Timer Expires()

3: Start L2 meas for PDCP - PjMiLpjL2MeasReq()

4: Start L2 meas for RLC - KwMiLkwL2MeasReq()

5: Start L2 meas for MAC - RgMiLrgSchL2MeasReq()

6: Send Request for Collecting Measurement()

7: PDCP L2 Meas. Confirmation - PjMiLpjL2MeasCfm()

8: Notification to OAM()

9: RLC L2 Meas. Confirmation - KwMiLkwL2MeasCfm()

10: Notification to OAM()

11: MAC L2 Meas. Confirmation - RgMiLrgSchL2MeasCfm()

12: Notification to OAM()

Figure-6: L2 Measurement Procedure

**Explanation**:

The steps for collecting the L2 measurement statistics are as follows.

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

## 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.

### 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. |

#### Sequence Diagram for Reporting of Fault

OAM

OAM CL

eNB Stack Manager

eNB Application

1: wrAlarmEvt()

2: FapAlarmInd-Id, Severity,

Cause type, Cause value()

3: Notification to OAM()

Figure-7: Reporting of Fault Procedure

**Explanation**:

The steps for reporting and collecting the fault are as follows:

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

# Integrating OAM CL with eNodeB

The following are the generic steps that may be needed to integrate OAM CL Interface with HeNB binary.

1. OAM CL to be built as a library (with makefile(s) 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

# Features Supported

The OAM features supported are as follows:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Support** | **Remarks** |
| Static configuration | Yes | Parameters are described in the TeNB\_Supported\_Parameters\_2.0.xlsx document. |
| Parameter validation | Yes | Parameters are described in the TeNB TeNB\_Supported\_Parameters\_2.0.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 U-ARM. Intel support is required for L-ARM rebooting. |
| Dynamic updating | Yes | 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. Six Connection Requests |
| Command Line Interface (CLI) | Yes | Parameters are described in the TeNB\_Supported\_Parameters\_2.0.xlsx document. |
| Performance Management (PM) | Yes | L2 and L3 counters. |
| Fault Management (FM) | Yes | Supports fault generation (currently, 8 faults) only. |
| WatchDog Application (WDApp) | Yes | A combination of software and kernel watchdog. |

Refer to the TeNB\_Supported\_Parameters\_3.0.xlsx document for more details in the release package.



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