|  |
| --- |
|  |
| rsys_logo |
| **LTE TotaleNodeB Common Platform**  **API Definition**  **1100464 1.0** |
|  |
|  |
|  |

© 1998-2013 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 4](#_Toc346888702)

[1.1 Purpose 4](#_Toc346888703)

[1.2 Scope 4](#_Toc346888704)

[1.3 Target Audience 4](#_Toc346888705)

[1.4 References 4](#_Toc346888706)

[1.4.1 Abbreviations 4](#_Toc346888707)

[1.4.2 Standards 5](#_Toc346888708)

[1.5 Revision History 5](#_Toc346888709)

[2 HeNB Architecture 6](#_Toc346888710)

[2.1 Functional Block Diagram 6](#_Toc346888711)

[3 API 8](#_Toc346888712)

[3.1 Performance Management 8](#_Toc346888713)

[3.1.1 API for Performance Counters 8](#_Toc346888714)

[3.1.2 API for Performance Counters 9](#_Toc346888715)

[3.1.3 Calling Sequence 9](#_Toc346888716)

[3.2 Configuration Management 10](#_Toc346888717)

[3.2.1 API for Static Configuration 10](#_Toc346888718)

[3.2.2 APIs for Dynamic Updating 12](#_Toc346888719)

[4 References 16](#_Toc346888720)

Figures

[Figure 1: Functional Block Diagram 6](#_Toc346888727)

[Figure 2: KPI Updating 9](#_Toc346888728)

[Figure 3: Static Configuration Call Sequence 11](#_Toc346888729)

[Figure 4: Dynamic Updating Call Sequence 14](#_Toc346888730)

Table

[Table 1 Abbreviations 4](#_Toc346574247)

# Introduction

## Purpose

This document describes the API provided by Radisys TotaleNodeB (TeNB), which is required for integration with common platform.

## Scope

This document contains the API provided by TotaleNodeB for configuration management and performance management.

## Target Audience

The target audiences for this document are:

* Developers involved in TotaleNodeB solution
* System Validation of TotaleNodeB solution
* System Engineers of TotaleNodeB solution
* Architects of the TotaleNodeB solution

## References

### Abbreviations

The following table lists the abbreviation used in this document.

**Table 1 Abbreviations**

|  |  |
| --- | --- |
| **Acronym** | **Description** |
| OAM | Operation, Administration and Maintenance |
| CWMP | Common WAN Management Protocol |
| CM | Configuration Management |
| FM | Fault Management |
| PM | Performance management |
| HeMS | Home eNodeB Management System |
| CLI | Command Line Interface |
| KPI | Key Performance Indicators |
| RAN | Radio Access Network |
| REM | Radio Environment Map |
| RRM | Radio Resource Management |
| SSI | System Services Interface |
| IPsec | IP Security |
| SON | Self-Organizing Networks |
| MIB | Management Information Base |
| NTP | Network Time Protocol |
| FTP | File Transfer Protocol |
| TCP | Transmission Control Protocol |
| TCP/IP | Transmission Control Protocol / Internet Protocol |
| UDP | User Datagram Protocol |
| PHY | Physical Layer |
| FSM | Finite State Machine |
| NAS | Non-Access-Stratum |

### Standards

1. [TR-196], “Femto Access Point Service Data Model”, Issue 2, November 2011.
2. [TR-069], “CPE WAN Management Protocol”, Issue 1, Amendment 4, July 2011.

## Revision History

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

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Date | Author (s) | Description |
| 1.0 | January 21, 2012 | Rajaram | Conforms to TotaleNodeB release version 1.0 |
| 0.1 | November 15, 2012 | Platform Team | Draft version |

# HeNB Architecture

This section introduces the Common Platform functional blocks within Radisys HeMS system and the interfaces between them.

## Functional Block Diagram

The following figure gives an overview of Radisys HeNB solution.

**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

Common Platform Components

PHY

**REM/SON**

**FTP**

**3rd party IPSec Stack**

**REM scan request /results**

**Stack management**

**Figure 1: Functional Block Diagram**

The HeNB system consists of ‘*Trillium core stack components’* and *‘Common Platform components*’.

The *Trillium core stack components* comprises of:

* Trillium Call Control Applications and FSM
* 3GPP compliant Trillium eNodeB protocol stack
* Trillium Silicon-Specific Convergence Layer
* Third-party provided UDP/IP Stack and PHY

Common Platform components comprises of:

* Common Framework and Infrastructure components
* OAM FSM and control module
* TR-069 client module
* Command line interface module
* IPSec controller module
* NTP client
* Management Information Base module (Data Model repository)
* REM controller and SON module.

For details, see Radisys\_LTE\_Common\_Platform\_SwArch document [3].

# API

This section outlines the APIs provided by TotaleNodeB to support performance management and configuration management. Following terminologies are used in the following sections:

**OAM-Messenger:** Interface between OAM and Stack Manager.

**Stack Manager:** Interface between eNodeB Application and OAM.

**KPI ID:** Performance Management counter is called Key Performance Identifier (KPI). In KPI Module, each KPI is identified by a unique ID called KPI ID. KPI ID is an enum value. KPI IDs are defined in KpiTypes.h file.

For example, KPI\_ID\_UNSUCCESSFUL\_RRC\_CONN\_REEST, KPI\_ID\_SUCCESSFUL\_RRC\_CONN\_REEST, ...

*Naming Convention: KPI\_ID\_XXXX, where XXXX: Procedure Name.*

## Performance Management

This section describes the APIs used for Performance Management.

### API for Performance Counters

|  |  |
| --- | --- |
| **API Details** | **Description** |
| API Name | ***IncFapKpiByIntVal*** |
| Synopsis | This function is used to inform about KPI changes to the OAM |
| Includes | wr\_kpi.h |
| Syntax | ***IncFapKpiByIntVal (KPI id, incVal )*** |
| Arguments | ***KPI*** –ID Each KPI is identified as a unique ID  ***incVal*** - Increment step (in + or -) (integer value) |
| Description | This function is used to inform about KPI changes to the OAM. It accepts KPIID (For example, RRC.establishment.Sum) and increment step (in + or -). |
| Return Values | None. |
| Where to Use | eNodeB Application shall call this API to increment the counter and notify OAM. This API is provided by OAM. |

### API for Performance Counters

|  |  |
| --- | --- |
| **API Details** | **Description** |
| API Name | ***IncFapKpiByRealVal*** |
| Synopsis | This function is used to inform about KPI changes to the OAM |
| Includes | wr\_kpi.h |
| Syntax | ***IncFapKpiByRealVal (KPI id, incVal )*** |
| Arguments | ***KPI*** –ID Each KPI is identified as a unique ID  ***incVal*** - Increment step (real value) |
| Description | This function is used to inform about KPI changes to the OAM. It accepts KPIID (For example, DRB.IPThpDl.QCI1) and increment step (in + or -). |
| Return Values | None. |
| Where to Use | eNodeB Application shall call this API to increment the counter and notify OAM. This API is provided by OAM. |

### Calling Sequence



**Figure 2: KPI Updating**

1. Stack informs OAM messenger through *IncFapKpi*() when any event KPI is noted at the stack.
2. SMApp in turn forwards KPI info to KPICollector.
3. KPICollector segregates the KPI into KPI group and updates the KPI counter.

Example on how to use the API in eNodeB application as follows:

*// /\*Increment KPI for Attempted RRC Connection Establishments\*/*

*PRIVATE S16 wrUmmRrcSetupHdlr*

*{*

*.....// RRC Connection Setup received*

*IncFapKpi( KPI\_ID\_LTE\_RRC\_ATTCONESTAB\_SUM, INC\_KPI\_VALUE\_BY\_ONE );*

*..*

*}*

## Configuration Management

### API for Static Configuration

|  |  |
| --- | --- |
| **API Details** | **Description** |
| API Name | ***MsmEnodeBinitialCfgComplete*** |
| Synopsis | This API is used to inform the eNodeB application that MIB is populated with the configuration parameter. |
| Includes | wr\_msm\_common.h |
| Syntax | ***MsmEnodeBinitialCfgComplete()*** |
| Arguments | None. |
| Description | This API is used to indicate the eNodeB application to start configuration of each layer. Once the MIB is populated with the configuration parameters configured by HeMS, this API is called after the admin state is set to “unlocked” by HeMS. This is used only during initialization of eNodeB. |
| Return Values | None. |
| Where to Use | This API is provided by Stack Manager and used by OAM-Messenger. |

|  |  |
| --- | --- |
| **API Details** | **Description** |
| API Name | ***MsmConfigComplete*** |
| Synopsis | This API is used to inform the OAM Messenger that the layer configuration is completed. |
| Includes | wr\_msm\_common.h |
| Syntax | ***MsmConfigComplete ()*** |
| Arguments | None. |
| Description | eNodeB application after configuring different layers indicates the OAM-Messenger that configuration is complete. Dynamic updating is processed once the initial configuration is completed. |
| Return Values | None. |
| Where to Use | This API is provided by OAM-Messenger and used by Stack Manger. |

#### Call Sequence



**Figure 3: Static Configuration Call Sequence**

1. HeMS sends the configuration parameters to HeNB. The TR-069 client parses these values and interfaces with OAM to populate the MIB.
2. Once all the configuration parameters are populated, HeMS changes the admin state to “unlocked”.
3. Once the admin state is changed to “unlocked”, OAM-Messenger indicates the Stack Manager by using ***MsmEnodeBinitialCfgComplete()*** API.
4. Stack Manager configures each layer and gets conformation from each layer.
5. Once the configuration of layers is completed, Stack Manager calls ***MsmConfigComplete()*** API to indicate OAM-Messenger that initial configuration is completed.

#### Examples

The code snippet on how to use the APIs is as follows.

##### Example Of MsmEnodeBinitialCfgComplete()

If (Admin state is UNLOCKED)

{

  if( eNodeB initialization is not triggered)

  {

           if (Messenger initialization configuration is completed)

                {

***MsmEnodeBinitialCfgComplete()***;

                 }

        }

}

**Note**: The above implementation is on the OAM-Messenger side.

**Example Of *MsmConfigComplete()***

If (the layer configuration is completed)

{

/\* inform OAM-Messenger that the configuration is completed \*/

***MsmConfigComplete()***

}

**Note**: The above implementation is on the Stack Manager side.

### APIs for Dynamic Updating

|  |  |
| --- | --- |
| **API Details** | **Description** |
| API Name | ***MsmAdminStateChanged*** |
| Synopsis | This API is used to inform the eNodeB application about the change in Admin state. |
| Includes | msm\_common.h |
| Syntax | ***MsmAdminStateChanged*** (adminState) |
| Arguments | U32 adminState |
| Description | This API is used to indicate the eNodeB application that Admin state is changed. This is used for dynamic updating of parameters from HeMS. If there is a change in service affecting parameter, then this API is used to lock the Admin state and subsequently unlock it after the changes are done. |
| Return Values | None. |
| Where To Use | This API is provided by Stack Manager and used by OAM. |

|  |  |
| --- | --- |
| **API Details** | **Description** |
| API Name | ***MsmDynamicConfiguration*** |
| Synopsis | This API is used to inform the eNodeB application about the dynamic updating of the parameters. |
| Includes | msm\_common.h |
| Syntax | ***MsmDynamicConfiguration*** (Void \*cfg, U32 CfgType, bool Action) |
| Arguments | Void \* cfg – The configuration structure  U32 CfgType – The configuration type such as SIB configuration.  Bool Action – Whether the changes to be applied immediately or deferred. |
| Description | This API is used to inform eNodeB application about the dynamic changes in the configuration parameters. This API accepts the structure of updated parameters as a void \* and based on the parameters changed, the eNodeB application takes appropriate action. |
| Return Values | None. |
| Where To Use | This API is provided by Stack Manager and used by OAM-Messenger. |

#### Call Sequence



**Figure 4: Dynamic Updating Call Sequence**

1. If dynamic updating happens for any service affecting parameters, HeMS changes the admin state to “locked”.
2. OAM-Messenger shall call the ***MsmAdminStateChanged()*** API to request the eNodeB application to switch off the transmitter.
3. HeMS configures the dynamic parameters and OAM-Messenger calls the ***MsmDynamicConfiguration ()*** API to send the changed parameters to eNodeB application.
4. HeMS updates the admin state to “unlocked”, once the dynamic configuration is completed.
5. Once all the configuration parameters are populated, HeMS changes the admin state to “unlocked”.

#### Examples

**Example of** ***MsmAdminStateChanged()***

  If (Admin state is LOCKED)

  {

       If (eNodeB configuration is completed)

{

               /\*This scenario for Service affecting parameter update from HeMS, hence locked again\*/

                 MsmAdminStateChanged(LOCKED);

        }

}

   else

   {

              /\*Admin state is UNLOCKED*. \*/*

              if(eNodeB configuration is completed)

              {

                      MsmAdminStateChanged(UNLOCKED);

               }

    }

**Note**: The above implementation is on the OAM-Messenger side.

**Example of** ***MsmDynamicConfiguration()***

If (eNodeB initial configuration is completed)

{

If (admin state is “LOCKED”)

{

MsmSmmDynCfgReq(&lteCellCfgParams, SI\_LTE\_CELL\_CONFIG\_PARAMETERS, 0);

}

}

**Note:** The above implementation is on the OAM-Messenger side.

# References

Refer to these documents for more information.

1. FRS TotaleNodeB
2. Radisys\_TeNodeB\_Software\_Architecture
3. Radisys\_LTE\_Common\_Platform\_SwArch
4. FSRS Configuration Management
5. FSRS Performance Management



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