`

****

*ICD and Design Document*

*Server*

**Mobile CEM Labs LLC**

4006 Belt Line Road,   
Suite 115 Addison, Texas, 75001  
United States

TABLE OF CONTENTS

Contents -------------------------------------------------------------------------------------------Page No.

[1 Introduction 7](#_Toc346703634)

[1.1 Scope 7](#_Toc346703635)

[1.2 Audience 7](#_Toc346703636)

[1.3 How This Document is Organized 7](#_Toc346703637)

[2 Architecture Overview 9](#_Toc346703638)

[3 Interface Definition 11](#_Toc346703639)

[3.1 Introduction 11](#_Toc346703640)

[3.2 Message Information 11](#_Toc346703641)

[3.3 Message Information Details 14](#_Toc346703642)

[4 Design Document 15](#_Toc346703643)

[4.1 Functional View 15](#_Toc346703644)

[4.2 Execution View 17](#_Toc346703645)

[4.3 Implementation Details 19](#_Toc346703646)

This page intentionally left blank

Chapter

*1*

Introduction

1. Scope

This document scope is to describe the interface with UE application that collect the data from diagnostics and upload to server. The document also detail out the design and Interface between the Diag Library and JNI Application.

1. Audience
2. How This Document is Organized

| # | Chapter | Contents |
| --- | --- | --- |

Table 1: Chapter Overview

The acronyms and abbreviations used in this document are listed below

| Acronym | Explanation |
| --- | --- |

Table 2: Acronyms and Abbreviations

Chapter

*2*

Architecture Overview

This section provide the reference of the module where its fits into complete echo system. The document does not intent to provide exact architecture of complete UE because this is not under the scope.



The Android phone provides multiple application for Drive test and walk test purpose. These application communicate interact with single application via service module. The service module interfaces with JNI Library to communicate with Qualcomm diagnostics port. The diagnostics library perform following task

* Interface with Diag Driver - Open , Send , Receive
* Enable/Disable the Logging and Event Information
* Asynchronously receive messages/event from the device
* Decode the messages
* Write to the log

Chapter

*3*

Interface Definition

* 1. Introduction

The interface between the DiagLibrary and JNI libray is “C” API. These interfaces expose initialization for Diagnostic Port, Enable/Disable of Logging Messages, Start/End of Logging file and provide asynchronous events to the application. This interface also provides the cell, neighboring cell, call – voice and packet data session relate information.

* 1. Message Information

These are functionality application interface are being expose by library.

**1.2.1> int init\_diag\_port( unsigned int manufacture , unsigned int systype, char\* file,int len, void (\*callback)(int, int , char \* , unsigned int ));**

**Purpose:** This function opens the device driver and setup the local data structure for the library

**Output Parameter:**  On successful, it shall return 0 and failure it shall return -1

**1.2.2> int set\_logging\_mask( int systype, int mask);**

**Purpose:** This function used to set up the logging mask to device.

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.3> int get\_logging\_mask (int systype);**

**Purpose:** This function for getting response for logging mask.

**Output parameter:** On successful, it returns 0 and failure it return -1.

**1.2.4> int set\_event\_mask( int mask);**

**Purpose:** This function for setting event mask to device.

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.5> int get\_event\_mask ( );**

**Purpose:** This function used for getting response of event mask.

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.6> int set\_uarfn\_value (unsigned short frequency);**

**Purpose:** This function sets the UARFCN frequency to device.

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.7> int close\_diag\_port();**

**Purpose:** This function used to close the device Diag-port

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.8> bool is\_freq\_set\_success();**

**Purpose:** This function used for checking device latching on desired UARFCN frequency.

**Output Parameter:** On successful, it returns true and failure it returns false (only for WCDMA).

**1.2.9> unsigned short get\_uarfn\_value();**

**Purpose:** This function gets the latched UARFCN frequency value on device. It returns the frequency set by “setUarfcnValue” function.

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.10> int get\_GSM\_CellInfo(GSMCellInfo \*gsmCellInfo );**

**Purpose:** This function gets GSM RRC information

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.11> int get\_WCDMA\_CellInfo (WCDMACellInfo \*wcdmacellInfo);**

**Purpose:** This function gets the WCDMA RRC information.

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.12> int get\_Lte\_CellInfo (LTECellInfo \*lteCellInfo);**

**Purpose:** This function gets the LTE RRC Information.

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.13> int start\_file\_writing(char\* imsi,char\* testid);**

**Purpose:** This function start the chipset data capturing process.

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.14> int stop\_file\_writing();**

**Purpose:** This function stops the data capturing process.

**Output Parameter:** On successful, it returns 0 and failure it return -1.

**1.2.15> void (\*callback)(int notifierType , int Identifier , char \* buf , unsigned int length ); // Api for callback registration.**

**Purpose:** This function used for making callback registration.

**Output Parameter:**

Where  
notifierType = LOGGING\_DATA/EVENT\_DATA.  
Identifier= LOG\_RRC\_MESSAGES/LOG\_RRC\_STATE\_MESSAGES/EVENT\_RRC\_STATE/...For viewing more identifiers refer “interface .h” file from Section 1.6.  
buf = Data buffer pointer.  
lenght= no of data bytes.  
manufacture = HTC/SAMSUNG.  
systype= GSM/WCDMA/LTE.  
mask= LOG\_RRC\_MESSAGES/LOG\_RRC\_STATE\_MESSAGES.... Refer to “interface .h” file from Section 1.6.  
GSMCellInfo= Structure for Gsm cell Information. For more info refer to “interface .h” file from Section 1.6.  
WCDMACellInfo=Structure for Wcdma cell Information. For more info refer to “interface .h” file from Section 1.6.  
LTECellInfo= Structure for Lte cell Information. For more info refer to “interface .h” file from Section 1.6.  
imsi =IMSI number.  
testid = 4 digit positive number.

* 1. Message Information Details



Chapter

*4*

Design Document

* 1. Functional View

The functionality includes five modules, device and global data structure. Each module has specific task to perform. Functional view diagram shows the communication of modules with each other.



The task of each module is given below.

**JNI Interface:** This is a JNI library that use the API expose by the DiagLibrary. The JNI Library provide interface to the application at Java API level.

**Interface module:** Interface module consist of API’s which are used to communicate with user level application. Once receiving function, it invoke specific module for their task.

**Diag module:** This module communicates with device drivers of device to open the device, close the device, write the data to the device and read the data from the device.

**DiagCmd module:**  These modules build the command that can be understood by Qualcomm chipset. This module implements the encoding of command like Logging- Enable, Logging-Disable, EFS parameter configuration etc. It uses Utility module to add CRC and padding bytes.

**Utility module**: This module utility functionality like generation and checking for CRC, padding bytes as per Qualcomm ICD.

**Decode module:** This module perform the decoding of message receive from devices. These convert it into the “C” structure format and update the global data structure. This also invokes the file writer module to write log message into the file.

**Log Module**: This module opens the file and writes into the file. The enabling of logging into the file would depend upon application. This module create the file with $IMSI\_$TESTID\_$SEQNUM whenever the file size is more than specified size, it shall close the open file and start the new file. Whenever an application stop the logging. It shall close the ongoing file.

**Global Data Structure:**

* 1. Execution View

The execution view, how functionality is executed in different physical model. The library module executes in two contexts

1. **Receiver Thread**

On library initialization, this module creates a thread which waits on Qualcomm Device Driver for receiving asynchronous response from the device. This thread handles the response of the messages which has been sent to devices, events occurs inside the devices and asynchronous logging messages received from the device.

1. JNI Library Context

Whenever application makes call, module executes the functionality either write the information on to the device or provide information from its global data structure.

The below represent the Flow Chart view of these functionality





* 1. Implementation Details

In source code implementation as per given in diagram below. JNI is top level directory in which all headers and ‘.c’ files are included.

1. 