Rui Huang

Designing Embedded Software Analysis System for Telecom Enterprise

Helsinki Metropolia University of Applied Sciences

Master’s Degree

Information Technology

Master’s Thesis

Date 02-05-2016

Author(s) Title Number of Pages Date	Rui Huang Designing Embedded Software Analysis System for Telecom Enterprise 49 pages + 1 appendices 2 May 2016
Degree	Master of Engineering
Degree Programme	Information Technology
Specialisation option	Mobile Programming
Instructor(s)	Kari Järvi, Principal Lecturer

In the recent mobile broadband world, the smart and powerful applications have been widely developed for consumer. Obviously, the most convincible instances are Apple's Mobile Operating System (iOS) and Google's Mobile Operating System (Android) applications. However, in the traditional industry, such as telecom industry, the mobile applications working as a part of a software analysis system have not yet been widely involved. The emergence of powerful mobile devices and cloud computing platforms has been driving the demand for a new solution to solve the complex industry problem.

The object of the thesis is to design and implement a primary mobile application system for supporting the maintenance work of analysing the embedded software status in telecom industry equipment. The system is designed on Server-Client model that consists of three functionality entities. The Mobile Client (MC) periodically calls the Cloud Platform API Provider (CPAP) and demonstrates the Device Under Test's (DUT) embedded software status; The CPAP is mainly implemented by Google App Engine (GAE); The DUT Server (DS) that acts as a bridge between the DUT and CPAP, retrieves the DUT's embedded software status and pushes the data to the CPAP. The DUT in the real world is telecom industry equipment.

In the present study, whole system architecture was designed, the CPAP and the MC were partially implemented, whereas the unimplemented feature will be continued with in the future.

Keywords	Google App Engine, Android, Python, Java, HTTP, SSH

Contents

Abbreviations

API	Application Programming Interface
Android	Google’s Mobile Operating System
CPAP	Cloud Platform API Provider
DUT	Device Under Test
DS	DUT Server
ESA	Embedded Software Analysis
GAE	Google App Engine
HAL	Hardware Abstraction Layer
HTTP	Hypertext Transfer Protocol
iOS	Apple’s Mobile Operating System
IPC	Inter-Process Communication
IPMB-L	Intelligent Platform Management Bus Local
IP	Internet Protocol
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MC	Mobile Client
OAuth	Open Standard for Authorization
R&D	Research & Design
SDK	Software Development Kits
SFTP	Secure File Transfer Protocol
SSH	Secure Shell Protocol
TCP	Transmission Control Protocol
UI	User Interface
UART	Universal Asynchronous Receiver / Transmitter
Windows Phone	Microsoft’s Mobile Operating System

1 Introduction

The thesis concentrates on general system architecture design and implementation of an Embedded Software Analysis (ESA) system for the telecom industry equipment that has three major function entities: The Mobile Client (MC), the Application Programming Interface (API) server that is named the Cloud Platform API Provider (CPAP), and the server connected to Device Under Test (DUT) named DUT Server (DS). The MC periodically retrieves the raw data from the CPAP and demonstrates all the statistics, it is a light-weight application. The CPAP communicates with the DS in the background, it maintains the accumulated DUT status information, including the hardware health information and the running embedded software status, performs all the analysis efficiently. The DS runs the Linux applications to get the DUT status via network services. Utilizing the whole system, the efficiency of the Research & Design (R&D) maintenance work can be heavily improved. The system is recommended to pilot in the sponsor company's R&D team widely in the future.

1.1 Background

In the recent telecom equipment industry, the R&D department still executes most of the maintenance tasks in a traditional way: Once there is a system catastrophe, the engineers have to access the equipment locally or remotely, collect all the massive symptom logs as well as all possible data with some dedicated designed maintenance service software. Normally, such software is complex and installed in a desktop or laptop environment. The software performs analysis locally on the computer and usually demands certain strict system requirements including hardware and software. For instance, it may require powerful Central Processing Unit (CPU) calculation capability, massive memory capacity, etc. In a word, such software is heavy-weight.

It is stated in Wikipedia that a mobile client application is a special computer program designed to run on the mobile devices, such as smartphones and tablets. [1] Due to emergence of higher capacity and much more powerful hardware performance, it is well understood that an intelligent powerful software can also be innovated and developed for the mobile devices, which is in the scope of mobile client application development.

According to an exclusive Automation World survey, applications handling specific industrial tasks are available for devices running Apple's Mobile Operating System (iOS), Google's Mobile Operating System (Android) and Microsoft's Mobile Operating System (Windows Phone), as well as common browsers. [2] The mobile broadband connectivity has been well developed, it makes sense and guarantees great performance in the area where the network accessibility is needed. Taking the advantage of such mobility, the R&D maintenance work can be triggered anywhere in real time, complies with the emergency rule. That means, the engineers can quickly and easily utilize the mobile like service solving the complex problem. In the telecom equipment industry, there is such demand for R&D maintenance work.

1.2 Technology Problem

Although the recent silicon technology provides a feasible solution of allowing the mobile devices running complex standalone tasks, from the end users' view point, the mobile client application should reach such a level as powerful as possible while light-weight. Firstly, the mobile devices can only have the limit power capacity, the client application must consume as little power as possible. Secondly, the mobile devices have the limit memory and storage, so, the client application must occupy as little hardware resource as possible.

Initially, the ESA system is specified to be a solo mobile client application, meaning the original draft system architecture design has only single mobile client connect the DUT directly, and it monitors the DUT's hardware health as well as the running embedded software status. Within this specification, the mobile client must get the massive data and store it temporarily and locally in the mobile device. In the meanwhile, the mobile client shall perform the complex analysis of the incoming data, updating the user interface continuously. The secure access to the DUT is also important for the user. After the feasibility study and group discussion within the participated Mobile Computing courses, it comes to a common understanding that all of the requirements above make the project technically challenging, and the mobile application is not able to achieve light-weight as the specification designs. A new system architecture design must be created for the project.

1.3 Object and Outcome

The object of the project is to design and implement an ESA system and its mobile client application must be light-weight. To achieve the object, the cloud computing platform has been taken into account, and also another dedicated server is utilized in the whole system architecture. Moving all heavy tasks including data management and analysis to the cloud computing platform makes the mobile client application more easily to develop to be light-weight.

The purpose of the thesis project includes the following:

- Creating an ESA system architecture
- Partial implementation of the ESA system, including the basic functionality of the CPAP and the light-weight MC application
- To see whether it is feasible and valuable to introduce and develop a mobile concept to the telecom equipment enterprise

1.4 Project Methodology

Firstly, the system architecture was designed. There were several mature technologies available for the project. Some of the technologies were evaluated and compared, and the best solution was figured out. Secondly, the whole project process was executed in software agile style, meaning the system requirement is flexible to modify according to the better understanding of the system, the software implementation can be modified and improved when the requirement is assessed and approved to modify. Thirdly, there were practical implementations and verification activities planned for each phase, the function unit test was performed when doing features implementation and system integration verification was planned when the whole system features are ready. Fourthly, the outcome of the project was reviewed and studied, the pros and cons are covered at the end of the study directing the project to reach a conclusion.

The project process steps are described here:

- The feasibility study for the project: for this step, the different technologies are evaluated, and primary system architecture is designed and created.
- The system architecture design and implementation specification design: for this step, the detailed system architecture is designed, and the implementation

specification covers the detail of each system entities, the specification is guideline for the following implementation activities.

- The system implementation: for this step, the main task is doing source coding for both the MC and the CPAP.
- The system verification: for this step, the main task is performing the test activities as planned, once there is bug found, correct it and redo the test.

1.5 Scope and Structure

The project is scoped to high-level system architecture design and system software implementation: the whole system architecture is fully designed, the system software is partially implemented, and some open topic is left for the future.

The thesis consists of five sections. Section 1 provides a brief introduction about the thesis project. Section 2 describes the theoretical and technical framework corresponding to the thesis project. Section 3 describes the designed system architecture, including the MC, the CPAP, the DS. Section 4 describes the practical implementation for the MC and the CPAP, some source code structures are also described in this section. Section 5 discusses and concludes the achievements and limitations of the project.

2 Theoretical and Technical Framework

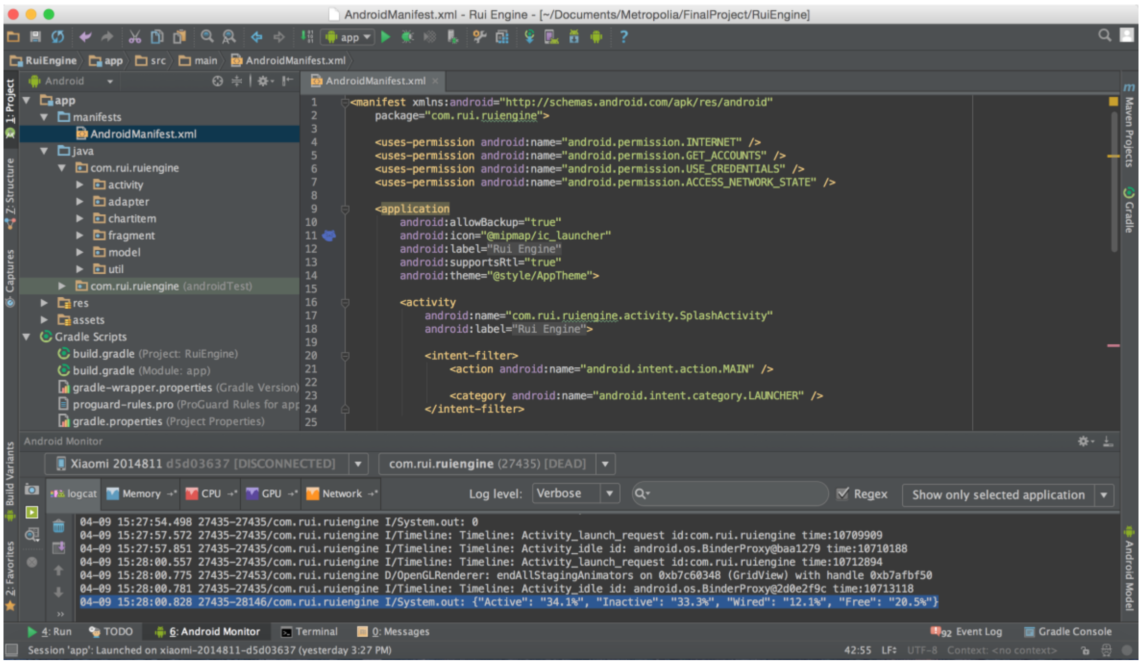
This chapter provides the description of the theoretical and technical details related to the design and implementation of the ESA System. It also provides a technical introduction related to the DUT system. *(Note: the DUT system is out of the scope of the system design)*

2.1 Mobile Client

The MC designed and implemented in the project provides a modern user interface and demonstrates all the requested information retrieved from the CPAP in the real time.

Worldwide, the Android, iOS, and Windows Phone are still the three biggest giants dominating the market share of the mobile smart devices operating system. Considering the mobile smartphone operating system shipments in Q3 of 2015, the Android was the strongest leader with 84.2% market share, while iOS and Windows came with the second and third, having 13.5% and 1.7% respectively. [3]

In the project, the Android mobile operating system was chosen due to the following reason: It offers a free, non-licensing and powerful development environment. In the project, the Android Studio was applied, Figure 1 shows its main Integrated Development Environment (IDE) window.


AndroidManifest.xml - Rui Engine - [~/Documents/Metropolia/FinalProject/RuiEngine]
RuiEngine
app
src
main
AndroidManifest.xml
Android
app
manifests
AndroidManifest.xml
java
com.rui.ruiengine
activity
adapter
chartitem
fragment
model
util
com.rui.ruiengine (androidTest)
res
assets
Gradle Scripts
build.gradle (Project: RuiEngine)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
Android Monitor
Xiaomi 2014811 d5d03637 [DISCONNECTED]
com.rui.ruiengine (27435) [DEAD]
logcat
Memory
CPU
GPU
Network
Log level: Verbose
Regex
Show only selected application
4: Run
TODO
6: Android Monitor
Terminal
0: Messages
92 Event Log
Gradle Console
Session 'app': Launched on xiaomi-2014811-d5d03637 (yesterday 3:27 PM)
42:55 LF: UTF-8 Context: <no context>

```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.rui.ruiengine">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.USE_CREDENTIALS" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Rui Engine"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name="com.rui.ruiengine.activity.SplashActivity"
            android:label="Rui Engine">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
04-09 15:27:54.498 27435-27435/com.rui.ruiengine I/System.out: 0
04-09 15:27:57.572 27435-27435/com.rui.ruiengine I/Timeline: Timeline: Activity_launch_request id:com.rui.ruiengine time:10709909
04-09 15:27:57.851 27435-27435/com.rui.ruiengine I/Timeline: Timeline: Activity_idle id: android.os.BinderProxy@baa1279 time:10710188
04-09 15:28:00.557 27435-27435/com.rui.ruiengine I/Timeline: Timeline: Activity_launch_request id:com.rui.ruiengine time:10712894
04-09 15:28:00.775 27435-27453/com.rui.ruiengine D/OpenGLRenderer: endAllStagingAnimators on 0xb7c60348 (GridView) with handle 0xb7afbf50
04-09 15:28:00.781 27435-27435/com.rui.ruiengine I/Timeline: Timeline: Activity_idle id: android.os.BinderProxy@2d0e2f9c time:10713118
04-09 15:28:00.828 27435-28146/com.rui.ruiengine I/System.out: {"Active": "34.1%", "Inactive": "33.3%", "Wired": "12.1%", "Free": "20.5%"}
```

Figure 1. Android Studio window

Android Studio is the official IDE for Android application development. It is based on IntelliJ IEDA. With IntelliJ's powerful code editor and developer tools, Android Studio provides several features such as flexible Gradle-based build system, build variants and multiple APK file generation, code templates, rich layout editor, lint tools, code shrinking, and built-in support for Google Cloud Platform. [4]

Also, the Android is open source operating system and there are plenty of open source libraries that help develop high quality applications.

The introduction of the Android system architecture and the release history are described in the following subsections.

2.1.1 Android System Architecture

According to Google's official statement, the Android system architecture contains a variety of software components, and has roughly seven sections and five layers as shown in Figure 2:

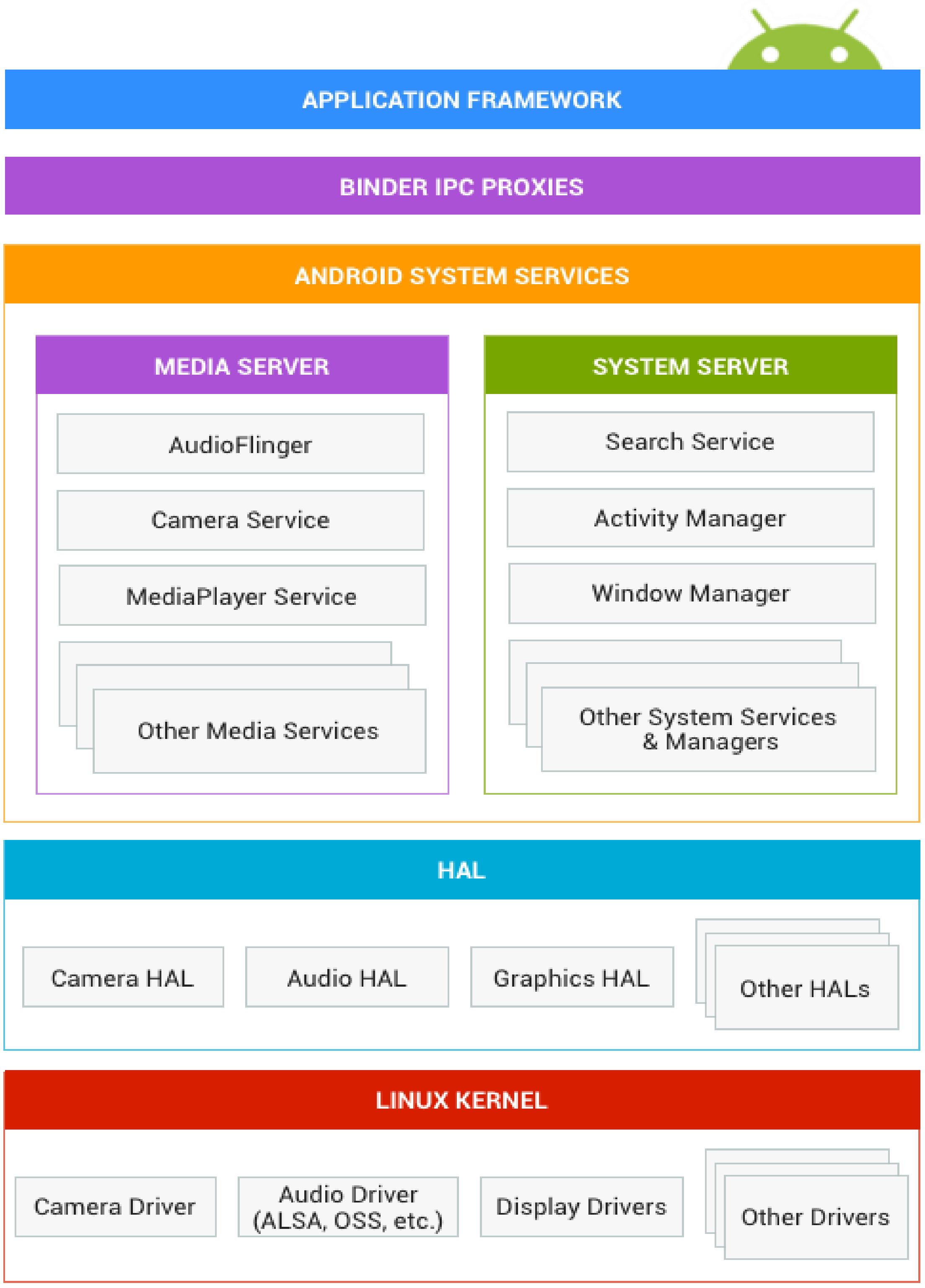


Figure 2. Android system architecture [5]

The five layers are: Application Framework, Binder Inter-Process Communication (IPC) Proxies, Android System Services, Hardware Abstraction Layer (HAL) and Linux Kernel.

Application Framework

The Application Framework layer provides the high-level services including the Activity Manager, the Content Providers, the Resource Manager, the Notifications Manager and the View System.

Binder IPC Proxies

When the Application Framework needs to cross the process boundaries, it must call the Binder IPC mechanism. The IPC is transparent to the developer at the Application Framework level.

Android System Services

The Android System Services are divided into two groups: the Media services and the System services. It allows functionality to access the underlying hardware resources.

Hardware Abstraction Layer

The Hardware Abstraction Layer is essential for different hardware vendors to implement hardware specific functionality for the Android system. Figure 3 shows an example of the Hardware Abstraction Layer (HAL).

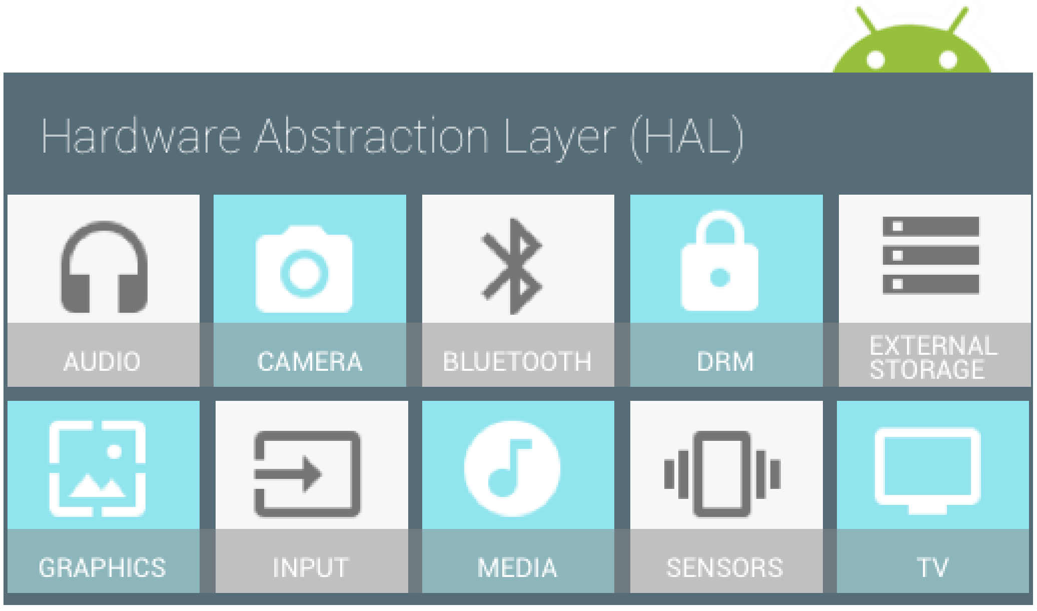


Figure 3. Android Hardware Abstraction Layer [5]

Linux Kernel

The Linux Kernel is at the bottom layer of the Android system architecture. It provides all types of hardware essential drivers, for instance, camera driver, audio driver, display drivers, etc.

In general, the Android system architecture is an end-to-end software architecture that allows the mobile operating system working with the different hardware configurations

2.1.2 Android Versions

The release history of the Android began with the release of the Android alpha in November 2007, and the most recent major Android update is Android 6.0 Marshmallow. [6] Figure 4 shows the global Android versions since December 2009.

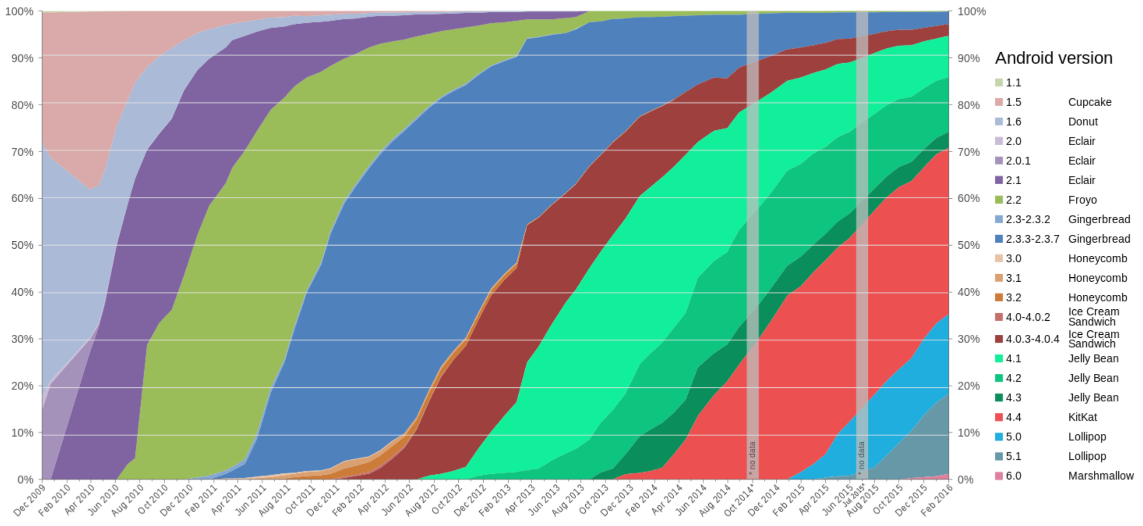


Figure 4. Global Android versions since December 2009 [6]

In the project, Android 5.1.1 Lollipop was chosen and the client application was built on it. The detail of the Android 5.1.1 Lollipop can be found in the following link: https://en.wikipedia.org/wiki/Android_Lollipop. (accessed December 18, 2015)

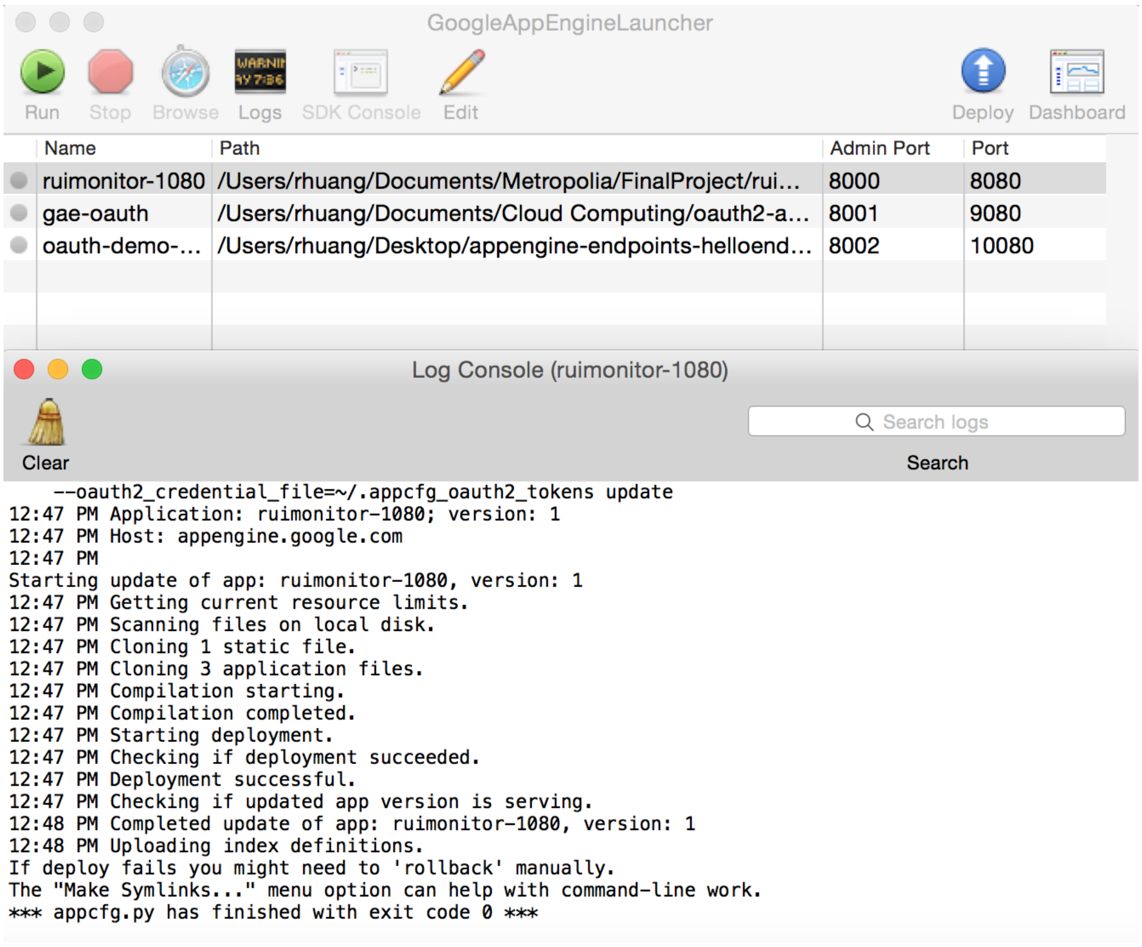
2.2 Cloud Platform API Provider

The CPAP designed and implemented in the project behaves mainly as the core of the whole system. The target DUT actively uploads or pushes back the predefined data to the CPAP. With these uploaded data, the CPAP also performs the Embedded Software

Status Analysis task according to the specified criteria. The CPAP provides the customized API for the MC to interact with. The project finally chose the Google App Engine (GAE) based on the Google Cloud Platform to comply with the system requirement. There are several reason for the decision: Firstly, the GAE provides all the features needed for the project, as explained in the next section. Secondly, there are a handful of open source references shared by the Google community and other organizations, they can be the very good design reference for the project. Last but not the least, the Google Cloud Platform provides a very friendly pricing policy and the Free Default Limit can fulfil the academic project purpose.

2.2.1 Google App Engine

The GAE provides ability for the developers to build the scalable web applications and the mobile backend. [7] The App Engine offers automatic scaling for web applications, as the number of requests increases for an application, the App Engine automatically allocates more resources for the web application to handle the additional demand. According to the cloud computing service definition, The GAE belongs to Platform-as-a-Service model abbreviated as the PaaS. The GAE can support a variety of programming languages including Java, Python, PHP and Go. [8] The developers can utilize its official Software Development Kits (SDK) to build a simulated environment that can emulate all the services on the local computer. With the deployment tools, developers can deploy the different versions of the application and manage the version control easily. Figure 5 shows the GAE SDK environment.



Name	Path	Admin Port	Port
ruimonitor-1080	/Users/rhuang/Documents/Metropolia/FinalProject/rui…	8000	8080
gae-oauth	/Users/rhuang/Documents/Cloud Computing/oauth2-a…	8001	9080
oauth-demo-…	/Users/rhuang/Desktop/appengine-endpoints-helloend…	8002	10080

```
    --oauth2_credential_file=~/.appcfg_oauth2_tokens update
12:47 PM Application: ruimonitor-1080; version: 1
12:47 PM Host: appengine.google.com
12:47 PM
Starting update of app: ruimonitor-1080, version: 1
12:47 PM Getting current resource limits.
12:47 PM Scanning files on local disk.
12:47 PM Cloning 1 static file.
12:47 PM Cloning 3 application files.
12:47 PM Compilation starting.
12:47 PM Compilation completed.
12:47 PM Starting deployment.
12:47 PM Checking if deployment succeeded.
12:47 PM Deployment successful.
12:47 PM Checking if updated app version is serving.
12:48 PM Completed update of app: ruimonitor-1080, version: 1
12:48 PM Uploading index definitions.
If deploy fails you might need to 'rollback' manually.
The "Make Symlinks..." menu option can help with command-line work.
*** appcfg.py has finished with exit code 0 ***
```

Figure 5. GAE deployment

As Google officially states, the GAE includes the following features:

- Persistent storage with queries, sorting and transactions.
- Automatic scaling and load balancing.
- Asynchronous task queues for performing work outside the scope of a request.
- Scheduled tasks for triggering events at specified or regular intervals.
- Integration with other Google cloud services and APIs. [9]

In the project, the CPAP was designed and partially implemented by several key features listed in Table 1.

Table 1. GAE features applied in the project cloud platform

App Identity	A framework that provides access to the application's identity, and the ability to assert this identity using OAuth
Channel	Creates a persistent connection between application and JavaScript clients in order to send messages in real time without polling
Google Cloud Endpoints	Generates APIs for Android, iOS, and web clients, making it easier to create a web backend for application
OAuth	Uses the OAuth protocol to provide a way for app to authenticate a user who is requesting access without asking for credentials (username and password)
Users	Allows an application to send and receive chat messages to and from any XMPP-compatible chat messaging service

With these key features, the CPAP meets the basic system requirements including data management, real time communication and security.

2.3 DUT System

The DUT system interacting with the ESA System is a hardware platform running the embedded Linux operating system. The DUT system provides the external interfaces so that it can connect via network.

From the software function modules point of view, the DUT system can be understood as three working modules, they are the Local Management Processor, the HW Management Controller, and the HW Management Target. The system hierarchy is shown in Figure 6.

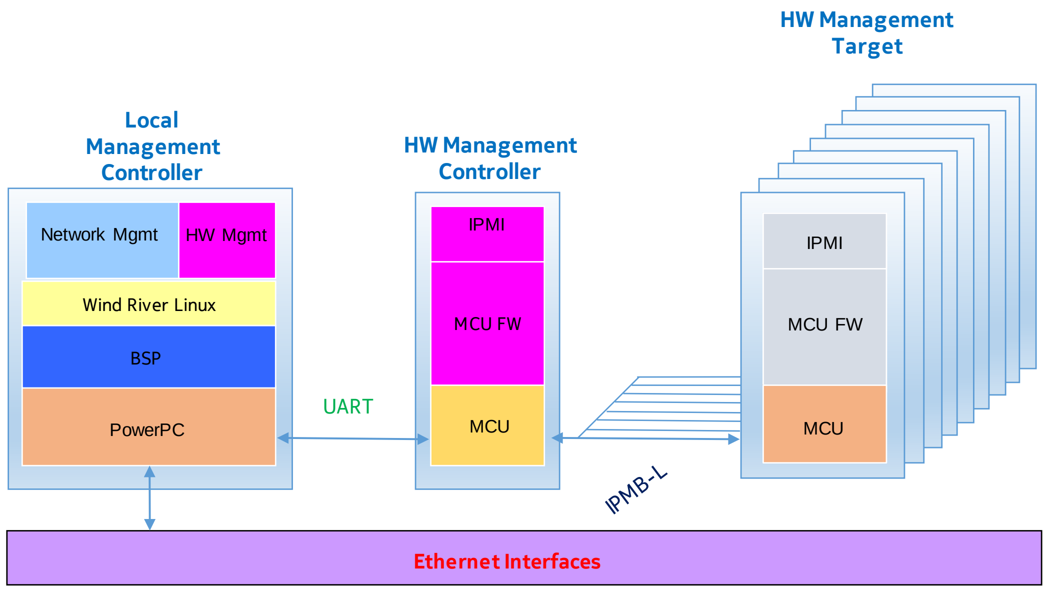


Figure 6. DUT System overview

The modules are linked with the universal asynchronous receiver/transmitter (UART) and Intelligent Platform Management Bus Local (IPMB-L) interfaces. The system messages can be communicated through these internal interfaces. Overall, the Local Management Controller is presenting to provide the necessary information for the ESA system. The modules' detailed responsibilities are described here:

- Local Management Controller

  The Local Management Controller provides the system software services on the basis of Wind River embedded Linux operating system. The applications running on the top of the operating system have the accessibility of the hardware health status and running software status. Through the Ethernet interfaces, the DUT can be connected from the outside network.
- Hardware Management Controller

  The Hardware Management Controller provides the hardware management services including on the local and remote sensors monitoring, the power payload control, etc.
- Hardware Management Target

  The Hardware Management Target is another entity mounted on the DUT system. It consists of several add-in cards units. The hardware health status can be read via IPMB-L interfaces.

The three modules are working together and generating the raw data required by the CPAP. In this case, once there is communication problem among modules, the raw data transmitted to the CPAP can be interrupted.

2.4 Communication Protocol

The CPAP designed for the project allows external entities uploading the data to the cloud data storage directly, the communication protocol running between the CPAP and external entities is the Hypertext Transfer Protocol (HTTP). The external entity involved in the project is the DS that connects DUT via the Secure Shell (SSH) protocol. The DS is actively retrieving the raw data from DUT, and the raw data will be pushed back to CPAP for the further handling. The CPAP provides the customized API to the MC to access its analysed data. The CPAP and the MC communicate via HTTP.

2.4.1 HTTP

The HTTP is an application protocol for distributed, collaborative, hypermedia information systems. [10] The protocol is built of the Request and the Response, and it is standard server-client model. Figure 7 illustrates the HTTP client and server communication.

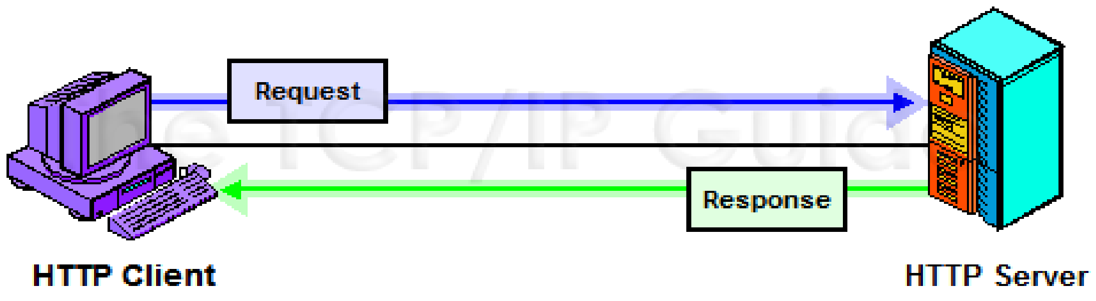


Figure 7. HTTP Client and Server communication [11]

The HTTP session is always initiated from the client side by generating a request to establish a Transmission Control Protocol (TCP) connection to a particular port on a server, typically the port is port 80. From the HTTP server side, it listens on that port and receives the client’s request message. When the server receives the request successfully, it replies a message to the client. [11]

The HTTP generic message format contains the Start Line, the Message Headers, the Empty Line Message Body and the Message Trailers. For the HTTP request and response message, they are slightly different.

Table 2 shows the structural elements of an HTTP request and an example of the sorts of headers a request might contain. [12]

Table 2. HTTP Request Message format

GET /index.html HTTP/1.1	Request Line	HTTP Request
Date: Thu, 20 May 2004 21:12:55 GMT Connection: close	General Headers	
Host: www.myfavoriteamazingsite.com From: joebloe@somewebsitesomewhere.com Accept: text/html, text/plain User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)	Request Headers	
	Entity Headers	
	Message Body	

Similar to the request message, the response message uses its own format as shown in Table 3. [13]

Table 3. HTTP Response Message format

HTTP/1.1 200 OK	Status Line	HTTP Response
Date: Thu, 20 May 2004 21:12:58 GMT Connection: close	General Headers	
Server: Apache/1.3.27 Accept-Ranges: bytes	Response Headers	
Content-Type: text/html Content-Length: 170 Last-Modified: Tue, 18 May 2014 10:14:49 GMT	Entity Headers	
<html> <head> <title>Welcome to the Amazing Site!</title> </head> <body>	Message Body	

```
<p>This site is under construction. Please come back later.
Sorry!</p>
</body>
</html>
```

The HTTP protocol provides the standardized method of network communication, it is quite frequently applied in the area of server-client application.

2.4.2 SSH

The SSH is a cryptographic (encrypted) network protocol operating at layer 7 of the Open System Interconnection model to allow the remote login and other network services to operate securely over an unsecured network. [14]

The SSH provides three major capabilities according to reference [15]:

- Remote command line login and remote command execution secured with SSH, known as Secure command shell
- File transfer using Secure File Transfer Protocol (SFTP)
- Port Forwarding.

In the project, the connection between the DS and DUT takes the File Transfer and Port Forwarding as the main support for its functionality implementation. The detail about the File Transfer and Port Forwarding is described here:

- File Transfer
  The SFTP is defined as a separate protocol layered over the SSH to handle file transfers. It encrypts both username/password and data being transferred. It uses the same port as the SSH server has, this helps eliminating the necessity of opening another port on the firewall or router. The Figure 8 shows an example about using SFTP on the extranet machines effectively restricts access to authorized users and encrypts usernames, passwords and files sent to or from the SSH server. [15]

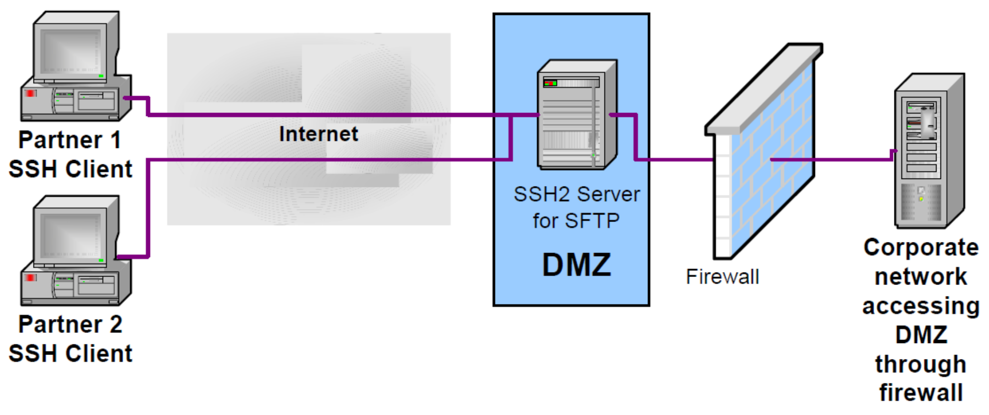


Figure 8. Single secure connection with multiple TCP/IP applications [15]

- Port Forwarding

  Port Forwarding provides sufficient remote control with graphical method and the encrypted tunnel over application is created. Figure 9 shows an example of a cross platform Graphical User Interface remote control application based on SSH Port Forwarding. [15]

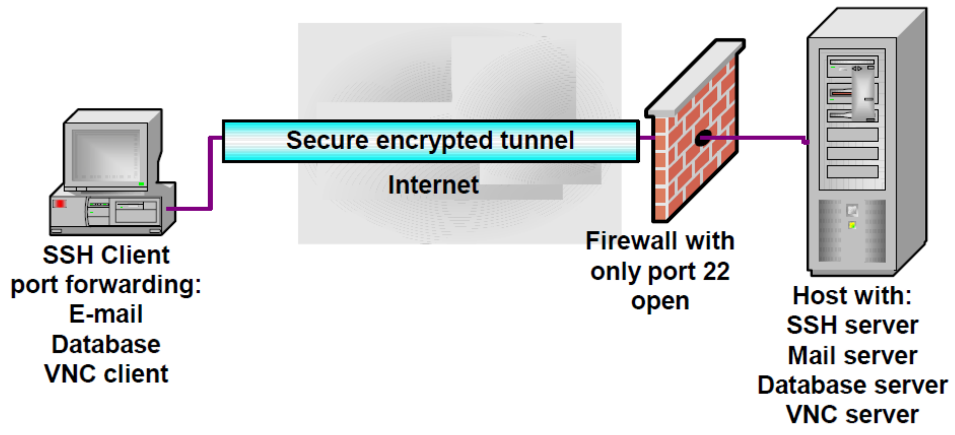


Figure 9. Single secure connection with multiple TCP/IP applications [15]

With the SSH protocol service, the communication between the DS and DUT is reliable and securable.

2.5 Security

The whole system connections are separated into three blocks: The MC and the CPAP are connected with the HTTP protocol; The CPAP and the DS are connected with the

HTTP protocol; The DS and the DUT are connected with the SSH protocol. The main security concern of the whole system for the project is the MC user access control to the CPAP. The Oauth2.0 is chose to provide secure access control, due to following reason: it is designed specifically to work with the HTTP. [15]

2.5.1 Oauth2.0

The Oauth2.0 is an open standard protocol that allows user authenticate the 3rd party website or application to access the information provided by another service provider. It doesn't require user to provide the user name and password to the 3rd party. [16] The Oauth2.0 protocol involves four participants:

- The Resource Owner
  It provides the authenticated right of accessing the resource.
- The Resource Server
  It stores the resource and deals with the request to access the resource.
- The Client
  It is the 3rd party application, when it is authenticated, it can access to the resource located in the Resource Server.
- The Authentication Server
  It authenticates the identity of the Resource Owner, provides authentication process for the Resource Owner, and delivers the access token.

In the Oauth2.0, the 3rd party client can be a website, or just a snip of Javascript code running in the web browser, or even a software application installed locally in the machine. There are specific security essences for different types of the 3rd party client. For the website application, it is isolated with the Resource Owner’s browser, it could save the sensitive information inside the protocol by its own effort, and the security key can be allowed secret to the Resource Owner. For the Javascript or local safe application, they are running inside of the Resource Owner, so, the Resource Owner can access to the sensitive client’s information inside the protocol.

The Oauth2.0 provides four different authentication type:

- Authorization Code Grant, it is applied to the application has its server backend
- Implicit Grant, it is applied to the pure Javascript client application
- Resource Owner Password Credentials Grant

- Client Credentials Grant.

Each authentication type is described in detail in the following part.

- Authentication Code Grant
  The Authentication Code Grant is applied for the PC or mobile client that needs communications with the 3rd party server, the process can be 2 steps, Step 1: the client app needs to acquire the authentication code; Step 2: the client app gets the access token by the authentication code. Figure 10 shows the detailed Authorization Code flow.

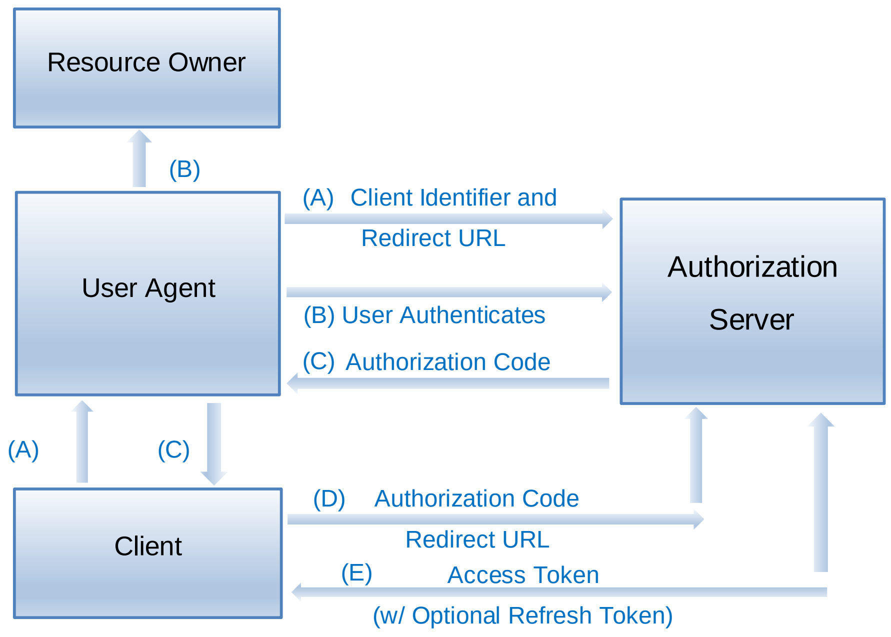

Resource Owner
(B)
User Agent
(A) Client Identifier and Redirect URL
(B) User Authenticates
(C) Authorization Code
Authorization Server
(A)
(C)
Client
(D) Authorization Code Redirect URL
(E) Access Token (w/ Optional Refresh Token)

Figure 10. Oauth2.0 Authorization Code Grant flow [17]

As Mohammad Amin Saghizadeh states, Oauth2.0 Authorization Code Grant flow can be illustrated as five steps [17]:

(A) When the client id, requested scope, local state and redirection URI are provided, the client triggers the flow to the Authorization Server. Whether the access is granted or denied, the Authorization Server can send back the received information.

(B) After the resource owner grants or denies the client’s access request, the Resource Owner is verified by the Authorization server, in addition the Authorization server establishes connection.

(C) When the Resource Owner grants access, the User Agent is sent back by the Authorization Server to the Client through the redirection URI provided earlier.

(D) When the Client makes a request of an access token from the Authorization Server's token endpoint, it sends the authorization code obtained from previous step. The Authorization Server and the Client verify this process.

(E) The Authorization Server is in charge of the Client authentication, the authorization code validation, and match between received redirection URI and URI mentioned in step (C). The authorization Server feedback an access token or a refresh token as option.

- Implicit Grant

  The Implicit Grant is normally applied in the client application that does not have server service. The client is the responsible entity to initiate the flow. For this type of grant, there is no returned fresh token. Figure 11 shows the detailed Implicit Grant flow.

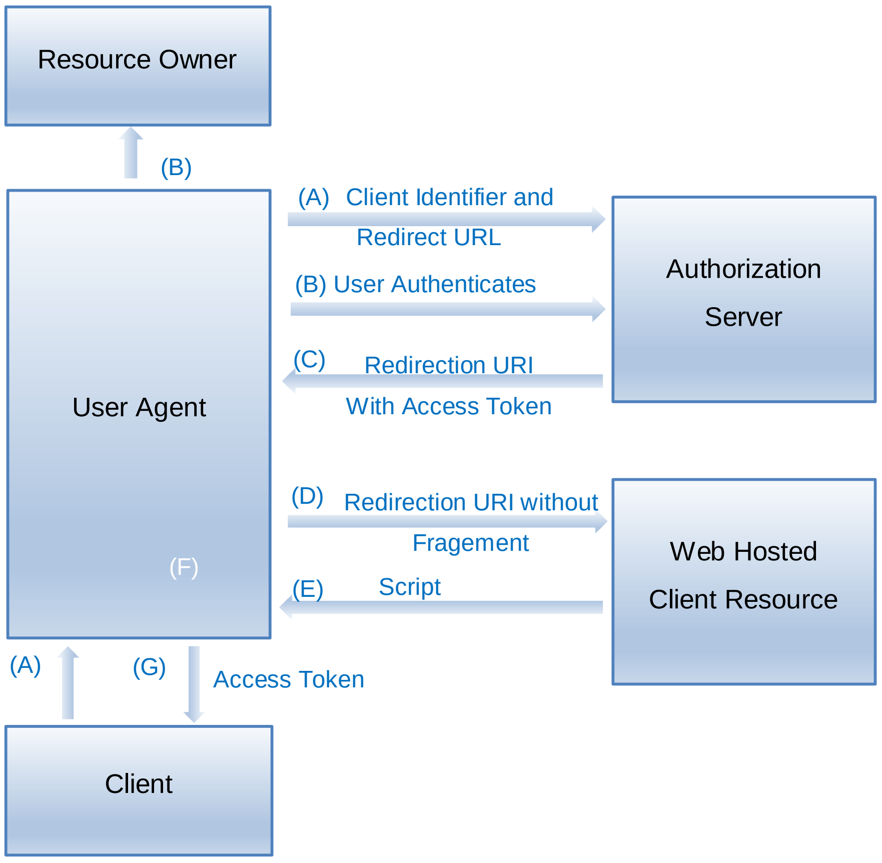

Resource Owner
(B)
(A) Client Identifier and Redirect URL
(B) User Authenticates
(C) Redirection URI With Access Token
Authorization Server
User Agent
(D) Redirection URI without Fragement
(E) Script
(F)
Web Hosted Client Resource
(A)
(G)
Access Token
Client

Figure 11. Oauth2.0 Implicit Grant flow [17]

This flow, it contains seven steps as show in reference article [17]:

(A) When the client id, requested scope, local state and redirection URI are provided, the client triggers the flow to the Authorization Server. Whether the access is granted or denied, the Authorization Server can send back the received information

(B) After the resource owner grants or denies the client's access request, the Resource Owner is verified by the Authorization server, in addition the Authorization server establishes connection.

(C) When the Resource Owner grants access, the User Agent is changed back by the Authorization Server to the Client through the redirection URI provided earlier.

(D) The User Agent makes a request to the web-hosted client resource to follow the redirection instructions. User Agent retains the fragment information locally.

(E) The Web Hosted Client Resource returns a web page, which is used to access the full redirection URI. This URI contains the fragment mentioned in step (D), and the access token (and other parameters) contained in the fragment is extracted by this URI.

(F) The local web-hosted client resource provides execution script and extracts the access token, and also the User Agent implements execution script.

(G) At last, the User Agent delivers the access token to the Client.

The theoretical and technical parts of the MC, CPAP and DUT are described above and two communication protocols, the HTTP and SSH are hereby explained. The Oauth2.0 is analysed for the purpose of security. In the next chapter, the detailed system architecture design is shown.

3 System Architecture

This chapter describes the system architecture design for the whole project. The whole system architecture is organized horizontally into three entities: the MC, the CPAP and the DS. Figure 12 illustrates the ESA system architecture.

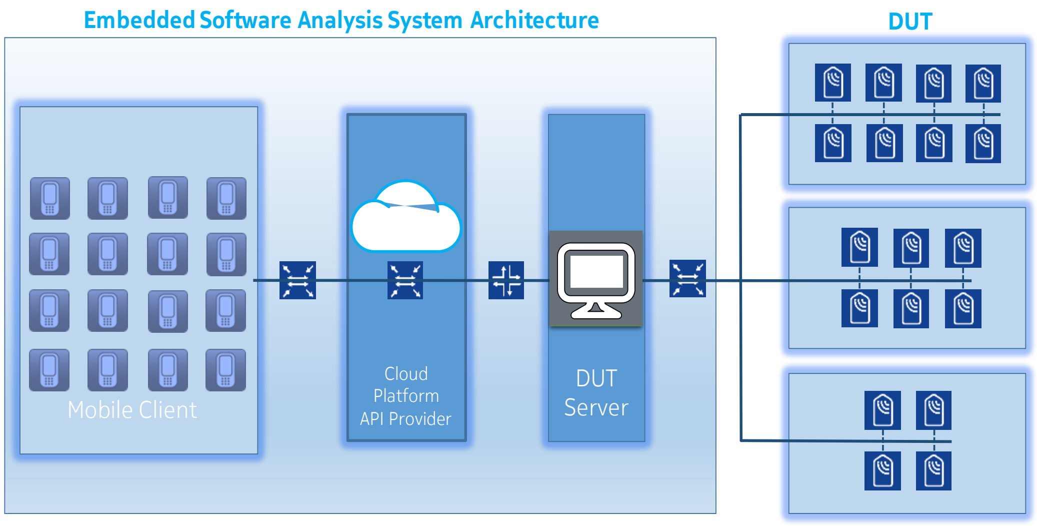

Embedded Software Analysis System Architecture
DUT
Mobile Client
Cloud Platform API Provider
DUT Server

Figure 12. ESA system architecture

The detail of the system architecture of is described in the following subsections.

3.1 Mobile Client System Architecture

The MC is booted with the Splash activity, it provides friendly information to the user for the first time of usage. The control of the user access is mainly under the Signin activity and Signup activity, the user needs to create new account for using the application. The legal and approved user account information is stored both locally in the MC and remotely in the CPAP. The MC communicates with the CPAP via the HTTP protocol, by sending the HTTP request, the MC receives the demanding data in the format of json. The information is interpreted in a vertical hierarchy, all the details can be referred to in Figure 13 below.

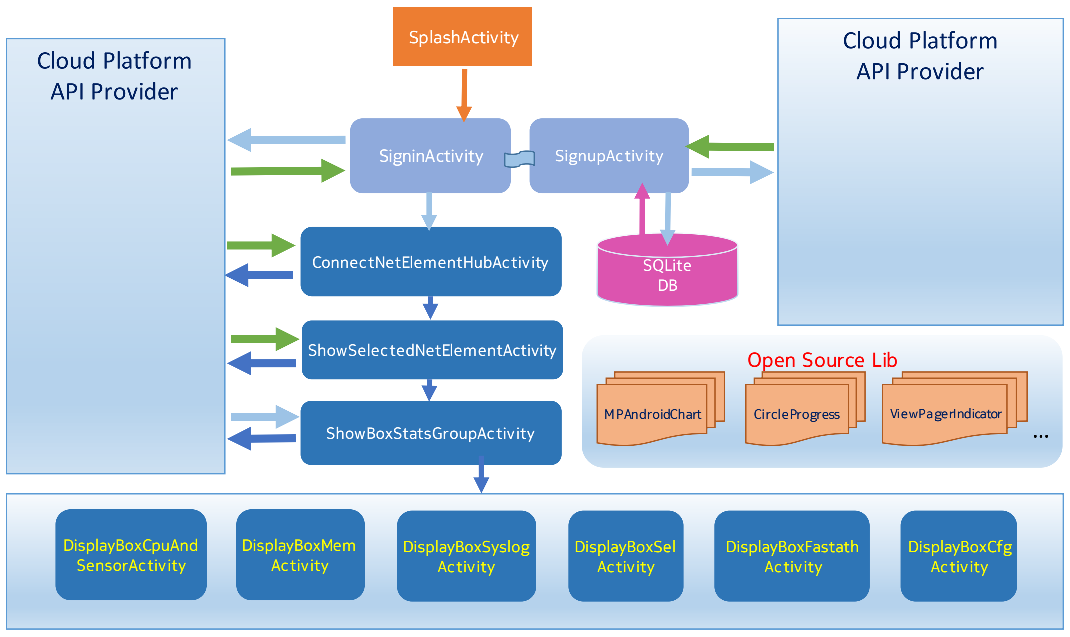


Figure 13. Mobile Client architecture

The MC application integrates several open source libraries to get the professional UI implemented.

3.2 Cloud Platform API Provider Architecture

The CPAP integrates the GAE as the main content API provider, according to the Google's recommendation, the Cloud Endpoints are selected to simply make direct API calls form the MC. [18] Figure 14 shows the abstract architecture for the CPAP.

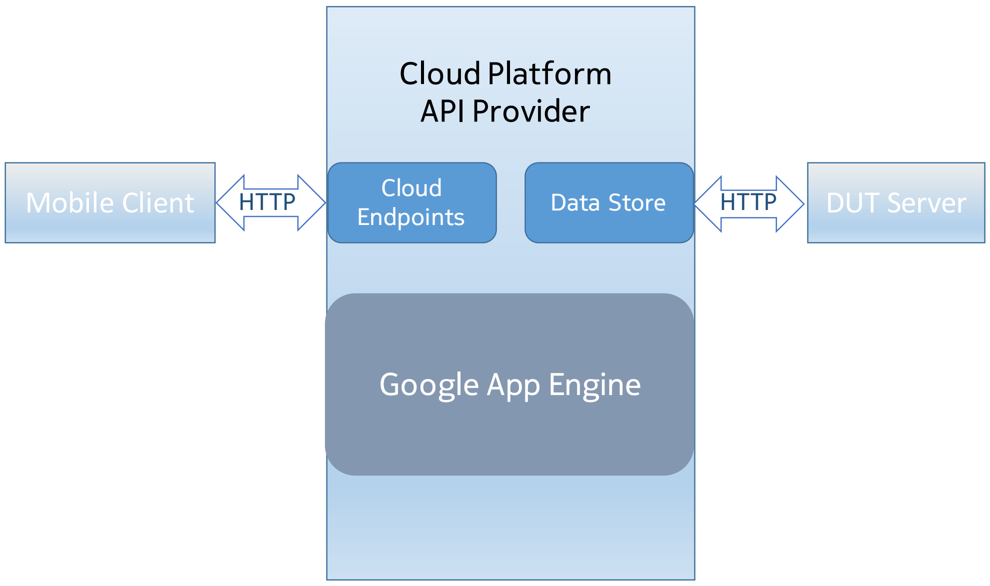


Figure 14. Cloud Platform API Provider architecture

The CPAP communicates with the MC and DS through HTTP ports, it is mainly implemented by the GAE. The GAE has two main units working, one is Cloud Endpoints that interacts with the MC, the other is Data Store that stores the raw data from the DS.

3.3 DUT Server Architecture

The DS application is responsible of retrieving the raw data from the DUT, and pushing the retrieved data to the Cloud Platform. The server application is built with the HTTP client and SSH server that provide external interfaces for data communication. The DS is a normal Linux operating system server that built with powerful personal computer or professional server. Figure 15 shows the DS software architecture.

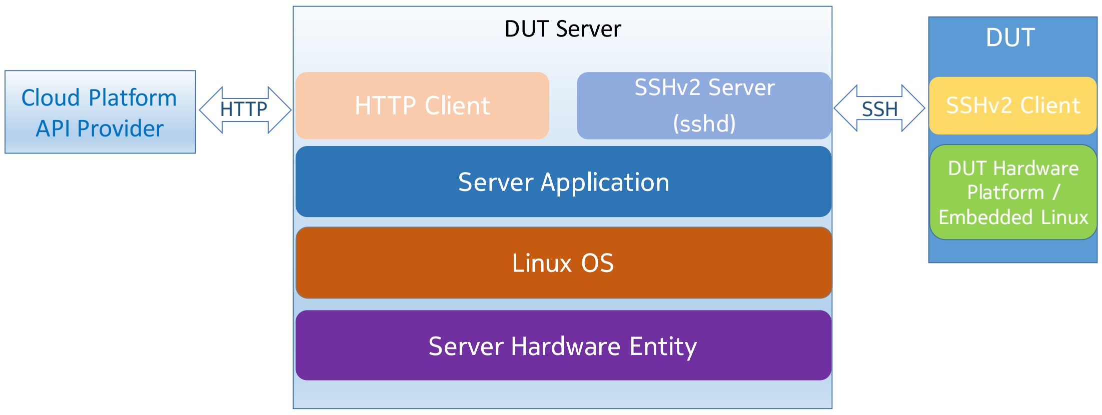


Figure 14. The DUT Server architecture

The system architecture provides the guideline of the implementation of each entity. Each entity performs a specific functionality, communicating and cooperating as a whole system. The details of the implementation are described in the next chapter.

4 Implementation

This chapter describes the detailed implementation for both the MC and CPAP. For the MC, the implementation follows the general operation flow chart, each operation is mapped to the corresponding Android activities. For the CPAP, the implementation covers the all steps about creating API engine in the Google Cloud Platform. An example of the demonstration is also shown in this section.

4.1 Mobile Client

The MC application is implemented with Android Lollipop 5.1, it provides the end users multiple interfaces to monitor the DUT's statistics and understand the running status with text and graphic information. The MC application consists of four main function units: Initialization, Access Control, Data Management, and Analytics. The overall structure is shown in Figure 16.

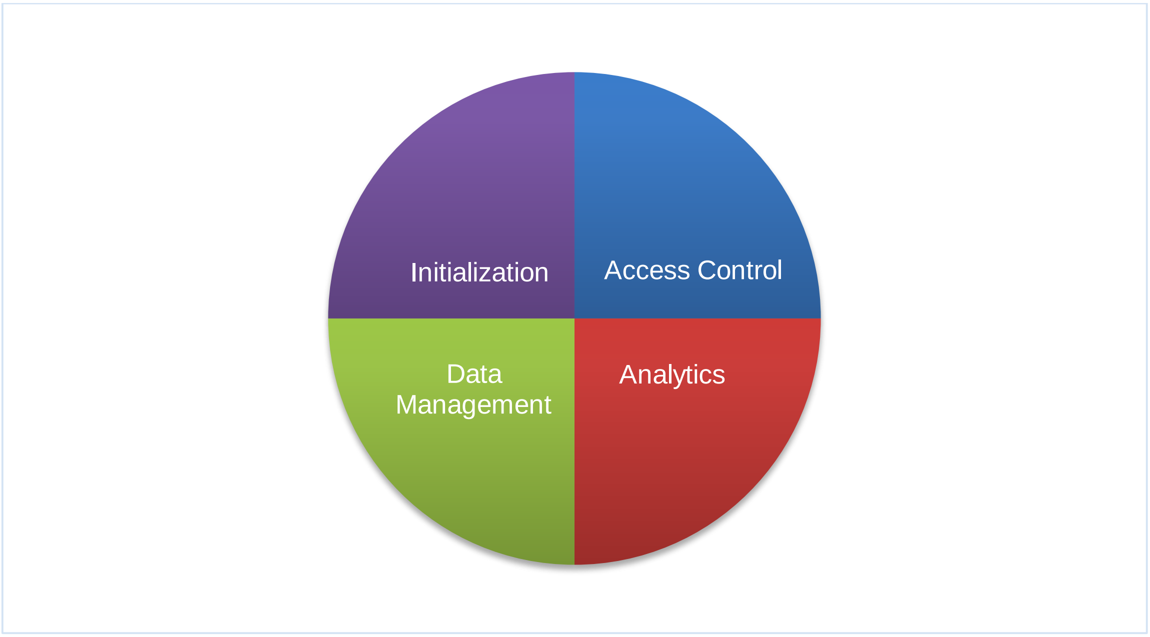


Figure 16. The Mobile Client function units overall structure

The real Android mobile device is utilized to run the MC application.

4.1.1 Initialization

When the MC application is loaded up, the app starts the Splash screen so that it provides the user the first impression that the application is professional and fancy. The Splash screen is created by the SplashActivity. A simple animation is implemented, which is taken the open source library and customized. The general working flow is shown in Figure 17.

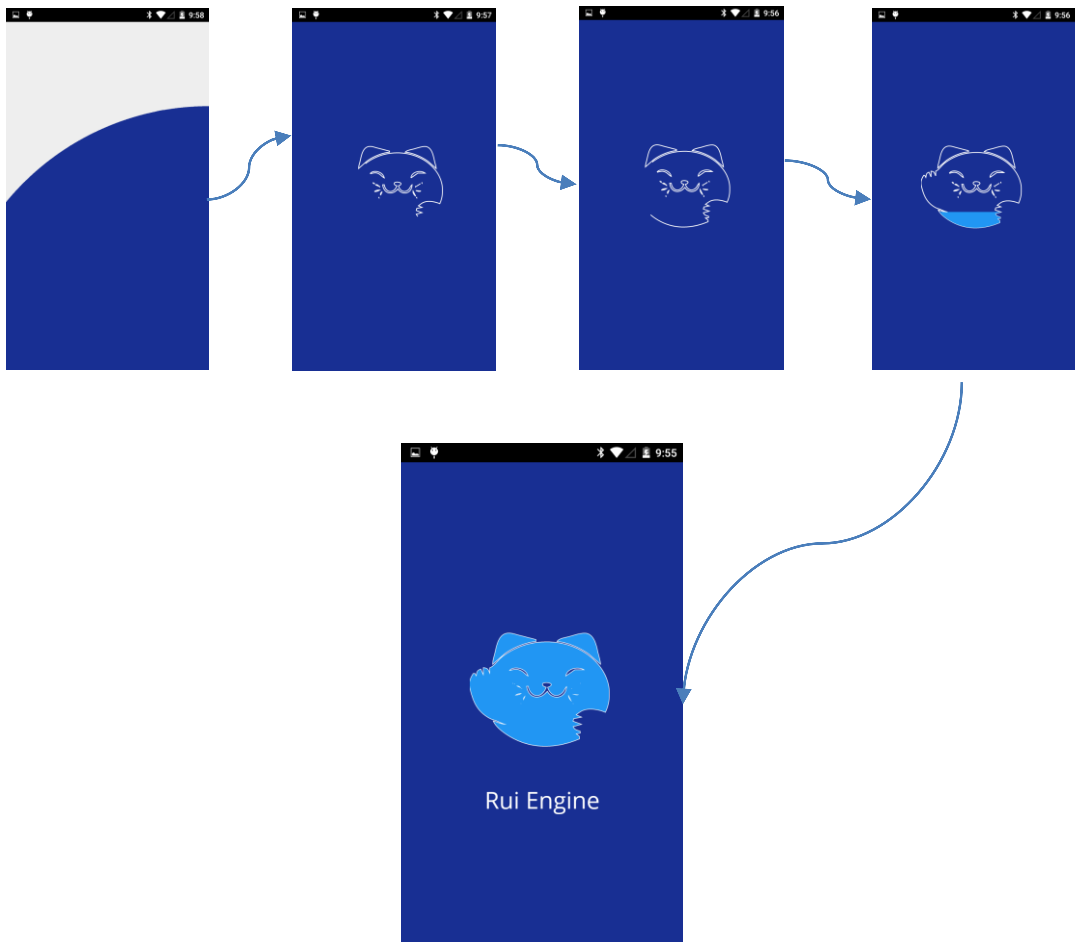


Figure 17. Mobile Client Splash screen

With the Splash screen finishes, the user can start using the MC. The Splash screen is handled by the SplashActivity, it is the first entry activity of the MC. The SplashActivity imports open source libraries:

```
com.nineoldandroids:library:2.4.0
com.daimajia.easing:library:1.0.1@aar
com.daimajia.androidanimations:library:1.1.3@aar
com.github.ViksaaSkool:AwesomeSplash:v1.0.0
```

And, the class hierarchy is shown as:

```
public class SplashActivity extends AwesomeSplash
{
    public void initSplash(ConfigSplash configSplash) {}
    public void animationsFinished() {}
}
```

When the Splash screen finishes, the application jumps to the MainActivity for the next step. It checks itself for the very first time started after the installation. There are two scenarios:

- The scenario 1: the first time using the application

  The MainActivity saves one software flag about the app has already been run for more than once. Then, it demonstrates four pages of view to the user so as some introduction about the app. In the last page, it provides the user an option button to start the real functionality of the Mobile Client. The general working flow is shown in Figure 18.

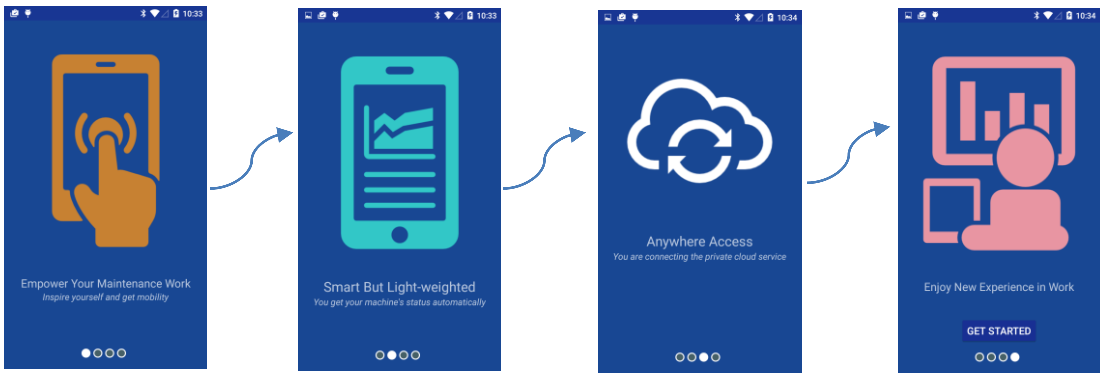

Empower Your Maintenance Work
Inspire yourself and get mobility
Smart But Light-weighted
You get your machine's status automatically
Anywhere Access
You are connecting the private cloud service
Enjoy New Experience in Work
GET STARTED

Figure 18. Mobile Client ViewPager screen

- The scenario 2: not the first time using the application

After finishing the SplashActivity, it goes directly to the SigninActivity. The general working flow is shown in Figure 19.

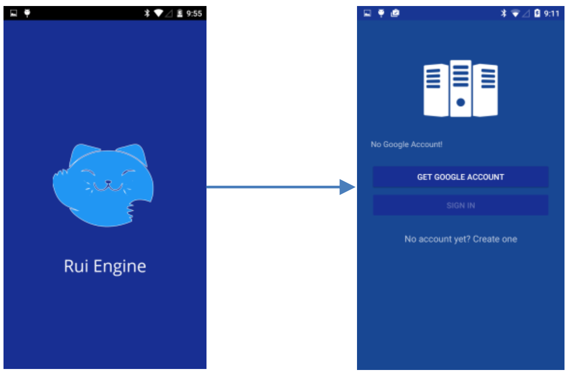


Figure 19. Mobile Client goes to sign in screen

The MainActivity handles the page navigation as the scenario 1 shows and jumps to the next step as the scenario 2 shows. The MainActivity imports open source library:

```
com.viewpagerindicator:library:2.4.1@aar
```

And, its class hierarchy is shown as:

```
public class MainActivity extends AppcompatActivity
{
    protected void onCreate(Bundle savedInstsanceState) {}
    protected void onPostResume() {}
    public static class SectionsPagerAdapter extends FragmentPagerAdapter
    {
        public Fragment getItem(int position) {}
        public int getCount() {}
    }
    public static class PlaceholderFragment extends Fragment
    {
         public static PlaceholderFragment newInstance(int sectionNum) {}
         public PlaceholderFragment() {}
    }
    public View onCreateView(LayoutInflater) {}
}
```

Overall, in the initialization phase, the code flow is briefly shown in Figure 20 below.

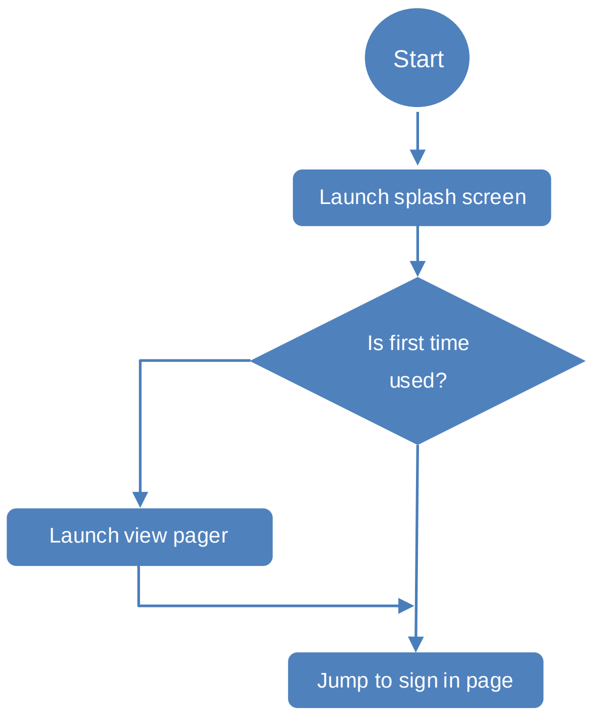


Figure 20. Mobile Client initialization phase

With the initialization phase done, the MC is ready to interact with the users.

4.1.2 Access Control

The MC access control is implemented with the Google account availability and the Google Play services via validating the application prior to allowing the OAuth2.0 access. Therefore, to use the Mobile Client, the user has to firstly register the mobile device with the available Google account in Google Play services. It can be achieved by signing the Google account as shown in the example in Figure 21.

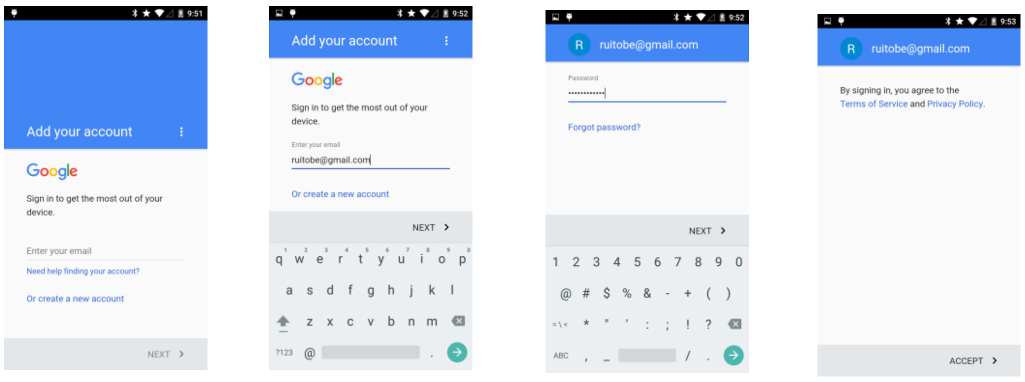
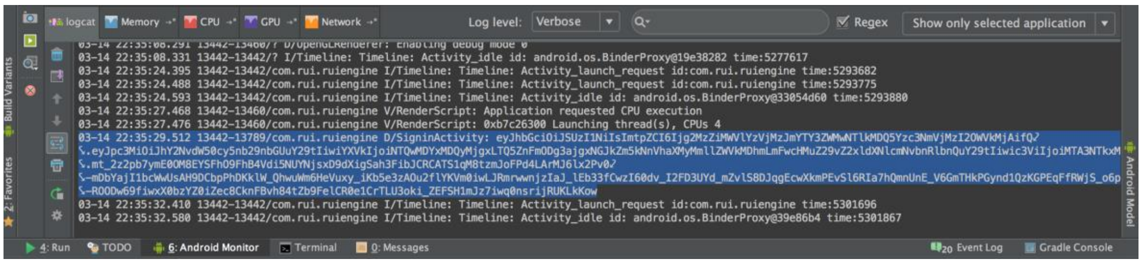


Figure 21. Mobile Client account service

Once the creating account is ready, from the sign in page, the user can click the "GET GOOGLE ACCOUNT" button to retrieve the registered Google account for the device. Once there is no Google account registered yet, an error pops up for a while. If the Google account is retrieved, the user clicks sign in button to get signed in.

In the background, the OAuth2.0 authentication process is running. If network access is available, the Mobile Client shall get the token as below, automatically, with the successful token, the Mobile Client will jump to the next stage. The highlighted part in the following figure is the token that the Mobile Client retrieves, see Figure 22.



```
03-14 22:35:08.291 13442-13460/? D/OpenGLRenderer: Enabling debug mode 0
03-14 22:35:08.331 13442-13442/? I/Timeline: Timeline: Activity_idle id: android.os.BinderProxy@19e38282 time:5277617
03-14 22:35:24.395 13442-13442/com.rui.ruiengine I/Timeline: Timeline: Activity_launch_request id:com.rui.ruiengine time:5293682
03-14 22:35:24.488 13442-13442/com.rui.ruiengine I/Timeline: Timeline: Activity_launch_request id:com.rui.ruiengine time:5293775
03-14 22:35:24.593 13442-13442/com.rui.ruiengine I/Timeline: Timeline: Activity_idle id: android.os.BinderProxy@33054d60 time:5293880
03-14 22:35:27.468 13442-13460/com.rui.ruiengine V/RenderScript: Application requested CPU execution
03-14 22:35:27.476 13442-13460/com.rui.ruiengine V/RenderScript: 0xb7c26300 Launching thread(s), CPUs 4
03-14 22:35:29.512 13442-13789/com.rui.ruiengine D/SigninActivity: eyJhbGciOiJSUzI1NiIsImtpZCI6Ijg2MzZiMWVlYzVjMzJmYTY3ZWMwNTlkMDQ5Yzc3NmVjMzI2OWVkMjAifQ↵
↘.eyJpc3MiOiJhY2NvdW50cy5nb29nbGUuY29tIiwiYXVkIjoiNTQwMDYxMDQyMjgxLTQ5ZnFmODg3ajgxNGJkZm5kNnVhaXMyMmllZWVkMDhmLmFwcHMuZ29vZ2xldXNlcmNvbnRlbnQuY29tIiwic3ViIjoiMTA3NTkxM
↘.mt_2z2pb7ymE0OM8EYSFhO9FhB4Vdi5NUYNjsxD9dXigSah3FibJCRCATS1qM8tzmJoFPd4LArMJ6lx2Pv0↵
↘-mDbYajI1bcWwUsAH9DCbpPhDKklW_QhwuWm6HeVuxy_iKb5e3zAOu2flYKVm0iwLJRmrwwnjzIaJ_lEb33fCwzI60dv_I2FD3UYd_mZvlS8DJqgEcwXkmPEvSl6RIa7hQmnUnE_V6GmTHkPGynd1QzKGPEqFfRWjS_o6p
↘-ROODw69fiwxX0bzYZ0iZec8CknFBvh84tZb9FelCR0e1CrTLU3oki_ZEFSH1mJz7iwq0nsrijRUKLkKow
03-14 22:35:32.410 13442-13442/com.rui.ruiengine I/Timeline: Timeline: Activity_launch_request id:com.rui.ruiengine time:5301696
03-14 22:35:32.580 13442-13442/com.rui.ruiengine I/Timeline: Timeline: Activity_idle id: android.os.BinderProxy@39e86b4 time:5301867
```

Figure 22. Oauth2.0 token

Also, the user can create a new Google account here, by clicking the text link "No account yet? Create one". It allows the user to call Google account service to create a new account. Note, for this page, the creation of the Google account is not yet implemented. See Figure 23.

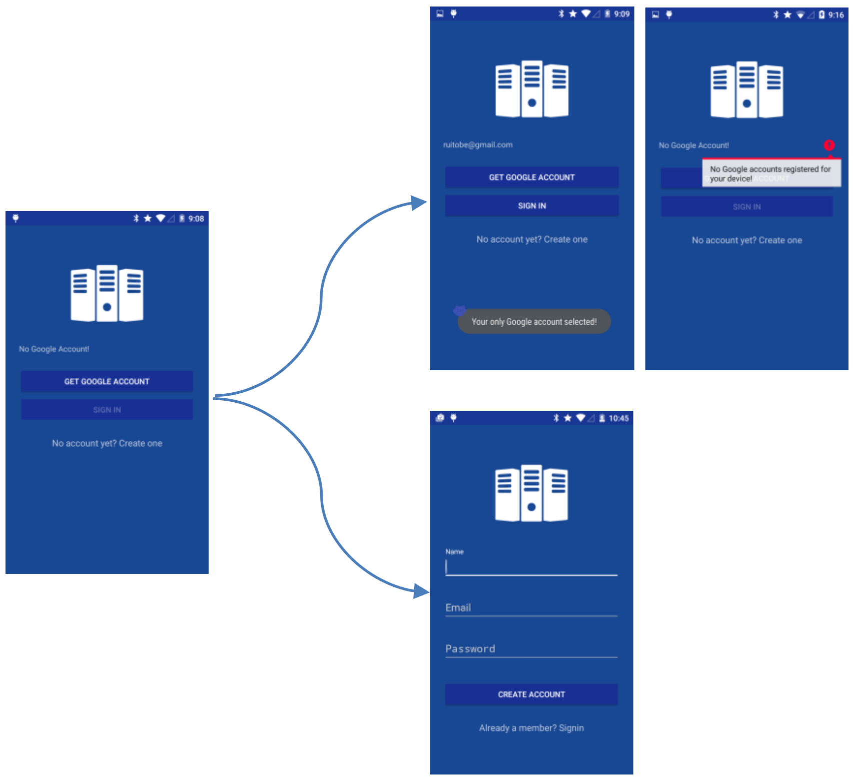


Figure 23. Mobile Client Signin and Signup

In this part, there are two activities involved, they are SigninActivity and SignupActivity. The SigninActivity imports open libraries:

```
group: ‘com.google.android.gms’, name: ‘play-services’, version: ‘3.2.25’
group: ‘com.google.guava’, name: ‘guava’, version: ’14.0.+’
group: ‘com.google.api-client’, name: ‘google-api-client-android’, version: ‘1.17.0-
rc’
```

And its class hierarchy shows as below:

```
public class SigninActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
    protected void onDestroy() {}
    protected void onActivityResult(int requestCode, int result code, intent data) {}
    public void onBackPressed() {}
    public void onLoginSucessful() {}
    public void onLoginFailed() {}
    public void getGoogleAccount() {}
    public void startAuthCheck(String account) {}
    public void signin() {}
    private class AuthorizationCheckTask extends AsyncTask<String, Integer, Boolean>
    {
       protected Boolean doInBackground(String… accounts) {}
       protected void onProgressUpdate(Integer… stringIds) {}
       protected void onPreExecute() {}
       protected void onPostExecute(Boolean aBoolean) {}
       protected void onCancelled() {}
    }
}
```

The SignupActivity class hierarchy is shown as below:

```
public class SignUpActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
    public void signUp() {}
    public void onSignupSuccessful() {}
    public void onSignupFailed() {}
    public Boolean validate() {}
}
```

The SigninActivity and SignupActivity provides a strict access control according to the system requirement.

4.1.3 Data Management

The MC retrieves the needed information mainly through the HTTP protocol, by sending the HTTP request to the CPAP. When the CPAP receives the request, it replies the MC the information in the JavaScript Object Notation (JSON) format. The information request is performed by the asynchronous task in the background. After the MC gets the JSON object, it interprets the JSON object and converts it to the string format, for the later usage. From the Android OS, the UI thread updates UI separately, in this case, the string converted from the JSON object will be displayed in the UI. Figure 24 shows the example.

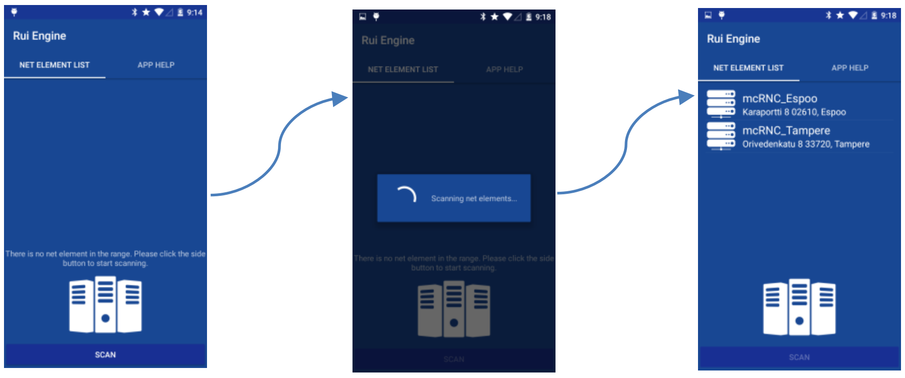
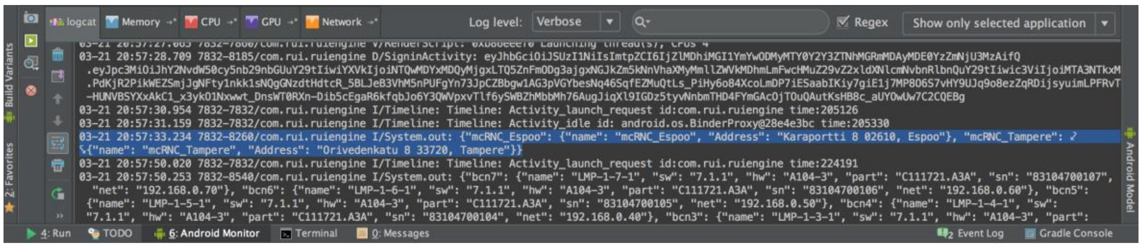

9:14
Rui Engine
NET ELEMENT LIST
APP HELP
There is no net element in the range. Please click the side button to start scanning.
SCAN
9:18
Rui Engine
NET ELEMENT LIST
APP HELP
Scanning net elements...
There is no net element in the range. Please click the side button to start scanning.
SCAN
9:18
Rui Engine
NET ELEMENT LIST
APP HELP
mcRNC_Espoo
Karaportti 8 02610, Espoo
mcRNC_Tampere
Orivedenkatu 8 33720, Tampere
SCAN

Figure 24. Mobile Client retrieves the site data

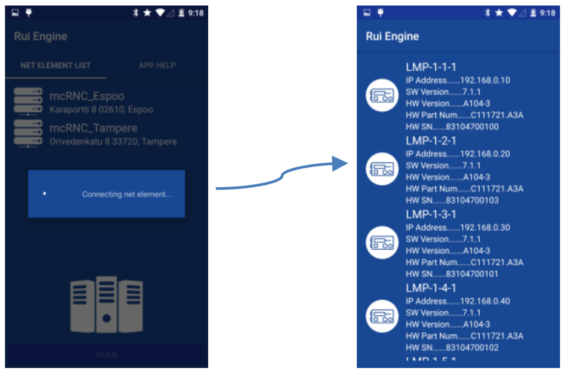
The user firstly gets an empty page by default. It is required to click the SCAN button to retrieve site information. If the network connection to the CPAP is working, the site information data will be retrieved and the MC updates the UI immediately. Otherwise, a failure message shows up. Figure 25 shows the site JSON format data successfully.



```
03-21 20:57:27.063 7832-7860/com.rui.ruiengine V/RenderScript: 0xb8eeee70 Launching thread(s), CPUs 4
03-21 20:57:28.709 7832-8185/com.rui.ruiengine D/SigninActivity: eyJhbGciOiJSUzI1NiIsImtpZCI6IjZlMDhiMGI1YmYwODMyMTY0Y2Y3ZTNhMGRmMDAyMDE0YzZmNjU3MzAifQ
 .eyJpc3MiOiJhY2NvdW50cy5nb29nbGUuY29tIiwiYXVkIjoiNTQwMDYxMDQyMjgxLTQ5ZnFmODg3ajgxNGJkZm5kNnVhaXMyMmllZWVkMDhmLmFwcHMuZ29vZ2xldXNlcmNvbnRlbnQuY29tIiwic3ViIjoiMTA3NTkxM
 .PdKjR2PikWEZSmjJgNFty1nkk1sNQgGNzdtHdtcR_5BLJeB3VhM5nPUFgYn73JpCZBbgw1AG3pVGYbesNq46SqfEZMuQtLs_PiHy6o84XcoLmDP7iESaabIKiy7giE1j7MP8O6S7vHY9UJq9o8ezZqRDijsyuimLPFRvT
 -HUNVBSYXxAkC1_x3ykO1Nxwwt_DnsWT0RXn-Dib5cEgaR6kfqbJo6Y3QWVpxvTlf6ySWBZhMbbMh76AugJiqXl9IGDz5tyvNnbmTHD4FYmGAcOjTOuQAutKsHB8c_aUYOwUw7C2CQEBg
03-21 20:57:30.954 7832-7832/com.rui.ruiengine I/Timeline: Timeline: Activity_launch_request id:com.rui.ruiengine time:205126
03-21 20:57:31.159 7832-7832/com.rui.ruiengine I/Timeline: Timeline: Activity_idle id: android.os.BinderProxy@28e4e3bc time:205330
03-21 20:57:33.234 7832-8260/com.rui.ruiengine I/System.out: {"mcRNC_Espoo": {"name": "mcRNC_Espoo", "Address": "Karaportti 8 02610, Espoo"}, "mcRNC_Tampere": ⤶
↳{"name": "mcRNC_Tampere", "Address": "Orivedenkatu 8 33720, Tampere"}}
03-21 20:57:50.020 7832-7832/com.rui.ruiengine I/Timeline: Timeline: Activity_launch_request id:com.rui.ruiengine time:224191
03-21 20:57:50.253 7832-8540/com.rui.ruiengine I/System.out: {"bcn7": {"name": "LMP-1-7-1", "sw": "7.1.1", "hw": "A104-3", "part": "C111721.A3A", "sn": "83104700107",
  "net": "192.168.0.70"}, "bcn6": {"name": "LMP-1-6-1", "sw": "7.1.1", "hw": "A104-3", "part": "C111721.A3A", "sn": "83104700106", "net": "192.168.0.60"}, "bcn5":
 {"name": "LMP-1-5-1", "sw": "7.1.1", "hw": "A104-3", "part": "C111721.A3A", "sn": "83104700105", "net": "192.168.0.50"}, "bcn4": {"name": "LMP-1-4-1", "sw":
 "7.1.1", "hw": "A104-3", "part": "C111721.A3A", "sn": "83104700104", "net": "192.168.0.40"}, "bcn3": {"name": "LMP-1-3-1", "sw": "7.1.1", "hw": "A104-3", "part":
```

Figure 25. Site data in JSON format

To go to the next step, the user clicks one of the list item. The MC navigates the user to the corresponding section when the network connection is working normally. Figure 26 shows the example.



mcRNC_Espoo
Karaportti 8 02610, Espoo
mcRNC_Tampere
Orivedenkatu 8 33720, Tampere

LMP-1-1-1
IP Address......192.168.0.10
SW Version......7.1.1
HW Version......A104-3
HW Part Num......C111721.A3A
HW SN......83104700100
LMP-1-2-1
IP Address......192.168.0.20
SW Version......7.1.1
HW Version......A104-3
HW Part Num......C111721.A3A
HW SN......83104700103
LMP-1-3-1
IP Address......192.168.0.30
SW Version......7.1.1
HW Version......A104-3
HW Part Num......C111721.A3A
HW SN......83104700101
LMP-1-4-1
IP Address......192.168.0.40
SW Version......7.1.1
HW Version......A104-3
HW Part Num......C111721.A3A
HW SN......83104700102

Figure 26. Mobile Client retrieves the element data

The data management is the essential part for the MC, there are the following activities involved in this part: the ConnectNetElementHubActivity and the ShowSelectedNetElementActivity. The ConnectNetElementHubActivity implements FragmentNetElementList.

The ConnectNetElementHubActivity class hierarchy shows as below:

```
public class ConnectNetElementHubActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
    private void setupViewPager(ViewPager viewPager) {}
}
```

FragmentNetElementList class hierarchy shows as below:

```
public class FragmentNetElementList extends Fragment
{
    public void onCreate(Bundle savedInstanceState) {}
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {}
    public void onViewCreated(View view, Bundle savedInstanceState) {}
    public void onActivityCreated(Bundle savedInstanceState) {}
    public void startRetrieveNetElementTask() {}
    public void scan() {}
    public boolean checkAvailability() {}
    public void onScanNetElementSuccessful() {}
    public void onScanNetElementFailed() {}
    public void connectNetElement() {}
    public void onConnectNetElementSuccessful() {}
    public void onConnectNetElementFailed() {}
    private class RetrieveNetElementsTask extends AsyncTask<Void, Void, String>
    {
        protected String doInBackground(Void… urls) {}
        protected void onPostExecute(String s) {}
    }
    public static String HttpRequest(int way, String url) {}
    public void startRetrieveNetElementsTask() {}
}
```

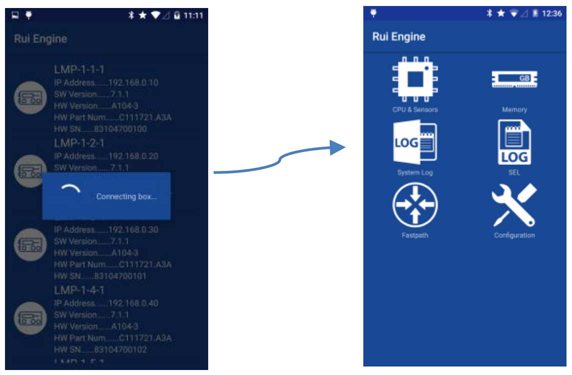
The ShowSelectedNetElementActivity class hierarchy shows as below:

```
public class ShowSelectedNetElementActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
    private class retrieveBoxesTask extends AsyncTask<Void, Void, String>
    {
        protected String doInBackground(Void… urls) {}
        protected void onPostExecute(String s) {}
    }
    public static String HttpRequest(int way, String url) {}
    public void startRetrieveBoxesTask() {}
    public void connectBox() {}
    public void onConnectBoxSuccessful() {}
    public void onConnectBoxFailed() {}
}
```

The next step is to select one target, the MC goes to the analysis part.

4.1.4 Analytics

The Analytics functionality unit gets the data analysis result and provides the user with a graphic view of the running target status. It provides six analytic items: the CPU & Sensors, the Memory, the System Log, the SEL, the Fastpath and the Configuration. All the data and information shown in the UI are real time data. The MC handles the data in the background thread and UI is updated by UI thread separately. There is one activity implemented in this section, it is ShowBoxStatsGroupActivity, running as an Analytics manager. There are six sub activities corresponding to the six analytic items: the DisplayBoxCpuAndSensorActivity, the DisplayBoxMemActivity, the DisplayBoxSyslogActivity, the DisplayBoxSelActivity, the DisplayBoxFastpathActivity, the DisplayBoxCfgActivity. See Figure 27.



```
LMP-1-1-1
IP Address......192.168.0.10
SW Version......7.1.1
HW Version......A104-3
HW Part Num......C111721.A3A
HW SN......83104700100
LMP-1-2-1
IP Address......192.168.0.20
SW Version......7.1.1
IP Address......192.168.0.30
SW Version......7.1.1
HW Version......A104-3
HW Part Num......C111721.A3A
HW SN......83104700101
LMP-1-4-1
IP Address......192.168.0.40
SW Version......7.1.1
HW Version......A104-3
HW Part Num......C111721.A3A
HW SN......83104700102
```

Figure 27. Mobile Client goes to analytics

The ShowBoxStatsGroupActivity class hierarchy shows as below:

```
Public class ShowBoxStatsGroupActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
}
```

The DisplayBoxCpuAndSensorActivity imports the open source library:

```
com.github.lzyzsd:circleprogress:1.1.0@aar
```

And the class hierarchy shows as below:

```
Public class DisplayBoxCpuAndSensorActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
    private void setupViewPager(ViewPager viewPager) {}
    private class RetrieveBoxCpuAndSensorTask extends AsyncTask<Void, Void, String>
    {
        Protected String doInBackground(Void… urls) {}
        Protected void onPostExecute(String s) {}
    }
    public static String HttpRequest(int way, String url) {}
    public void startRetrieveBoxCpuAndSensorTask() {}
}
```

The DisplayBoxCpuAndSensorActivity retrieves the data from Cloud Platform API provider, and updates the user interface as below (Figure 28):

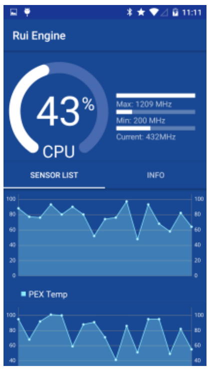
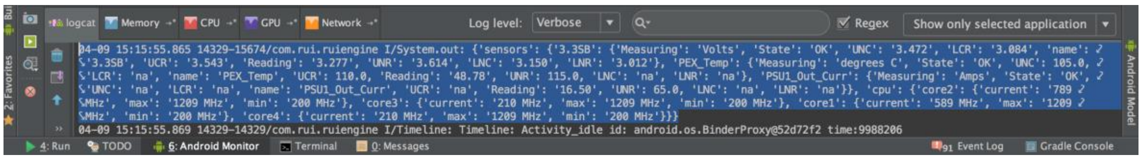


Figure 28. Mobile Client shows CPU and Sensors statistics

The MC shows the running CPU loading and speed, also all the sensors reading are shown in the list dynamic graphic charts. The MC performs the task in the background, sends the data request to the CPAP. If network is working normally, the CPU and sensors data are retrieved in the JSON format as shown in Figure 29 below.



```
04-09 15:15:55.865 14329-15674/com.rui.ruiengine I/System.out: {'sensors': {'3.3SB': {'Measuring': 'Volts', 'State': 'OK', 'UNC': '3.472', 'LCR': '3.084', 'name': 
'3.3SB', 'UCR': '3.543', 'Reading': '3.277', 'UNR': '3.614', 'LNC': '3.150', 'LNR': '3.012'}, 'PEX_Temp': {'Measuring': 'degrees C', 'State': 'OK', 'UNC': 105.0, 
'LCR': 'na', 'name': 'PEX_Temp', 'UCR': 110.0, 'Reading': '48.78', 'UNR': 115.0, 'LNC': 'na', 'LNR': 'na'}, 'PSU1_Out_Curr': {'Measuring': 'Amps', 'State': 'OK', 
'UNC': 'na', 'LCR': 'na', 'name': 'PSU1_Out_Curr', 'UCR': 'na', 'Reading': '16.50', 'UNR': 65.0, 'LNC': 'na', 'LNR': 'na'}}, 'cpu': {'core2': {'current': '789 
MHz', 'max': '1209 MHz', 'min': '200 MHz'}, 'core3': {'current': '210 MHz', 'max': '1209 MHz', 'min': '200 MHz'}, 'core1': {'current': '589 MHz', 'max': '1209 
MHz', 'min': '200 MHz'}, 'core4': {'current': '210 MHz', 'max': '1209 MHz', 'min': '200 MHz'}}}
04-09 15:15:55.869 14329-14329/com.rui.ruiengine I/Timeline: Timeline: Activity_idle id: android.os.BinderProxy@52d72f2 time:9988206
```

Figure 29. CPU and Sensors data in JSON format

The DisplayBoxMemActivity imports the open source library:

```
com.github.PhilJay:MPAndroidChart:v2.1.6
```

And the class hierarchy shows as below:

```
public class DisplayBoxMemActivity extends AppCompatActivity implements OnChart-
ValueSelectedListener
{
    protected void onCreate(Bundle savedInstanceState) {}
    protected void onValueSelected(Entry e, int dataSetIndex, Highlight h) {}
    protected void onNothingSelected() {}
    private SpannableString generateCenterSpannableText() {}
    private void setData(int count, float range) {}
    private class RetrieveBoxMemTask extends AsyncTask<Void, Void, String>
    {
        Protected String doInBackground(Void… urls) {}
        Protected void onPostExecute(String s) {}
    }
    public static String HttpRequest(int way, String url) {}
    public void startRetrieveBoxMemTask() {}
}
```

The DisplayBoxMemActivity retrieves the memory data from the CPAP, and updates the user interface as below (Figure 30):

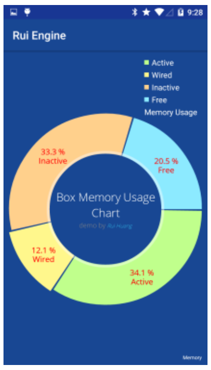
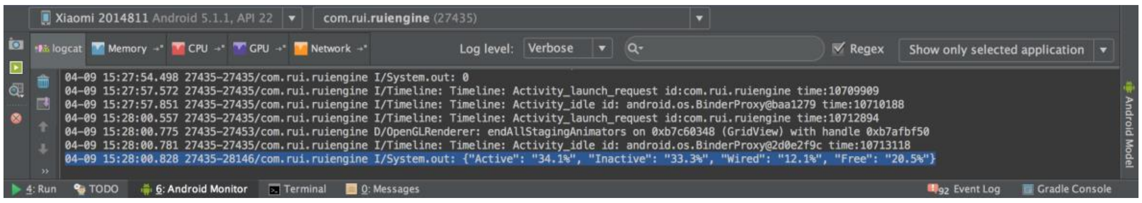


Figure 30. Mobile Client shows Memory statistic

If the network is working normally, the memory data is retrieved in the JSON format as shown in Figure 31 below.



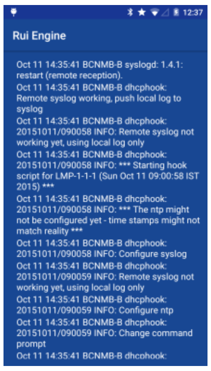
```
04-09 15:27:54.498 27435-27435/com.rui.ruiengine I/System.out: 0
04-09 15:27:57.572 27435-27435/com.rui.ruiengine I/Timeline: Timeline: Activity_launch_request id:com.rui.ruiengine time:10709909
04-09 15:27:57.851 27435-27435/com.rui.ruiengine I/Timeline: Timeline: Activity_idle id: android.os.BinderProxy@baa1279 time:10710188
04-09 15:28:00.557 27435-27435/com.rui.ruiengine I/Timeline: Timeline: Activity_launch_request id:com.rui.ruiengine time:10712894
04-09 15:28:00.775 27435-27453/com.rui.ruiengine D/OpenGLRenderer: endAllStagingAnimators on 0xb7c60348 (GridView) with handle 0xb7afbf50
04-09 15:28:00.781 27435-27435/com.rui.ruiengine I/Timeline: Timeline: Activity_idle id: android.os.BinderProxy@2d0e2f9c time:10713118
04-09 15:28:00.828 27435-28146/com.rui.ruiengine I/System.out: {"Active": "34.1%", "Inactive": "33.3%", "Wired": "12.1%", "Free": "20.5%"}
```

Figure 31. Memory data in JSON format

The DisplayBoxSyslogActivity class hierarchy shows as below:

```
public class DisplayBoxSyslogActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
    private class RetrieveBoxSyslogTask extends AsyncTask<Void, Void, String>
    {
        Protected String doInBackground(Void… urls) {}
        Protected void onPostExecute(String s) {}
    }
    public static String HttpRequest(int way, String url) {}
    public void startRetrieveBoxSyslogTask() {}
    private List<SyslogInfo> createList() {}
}
```

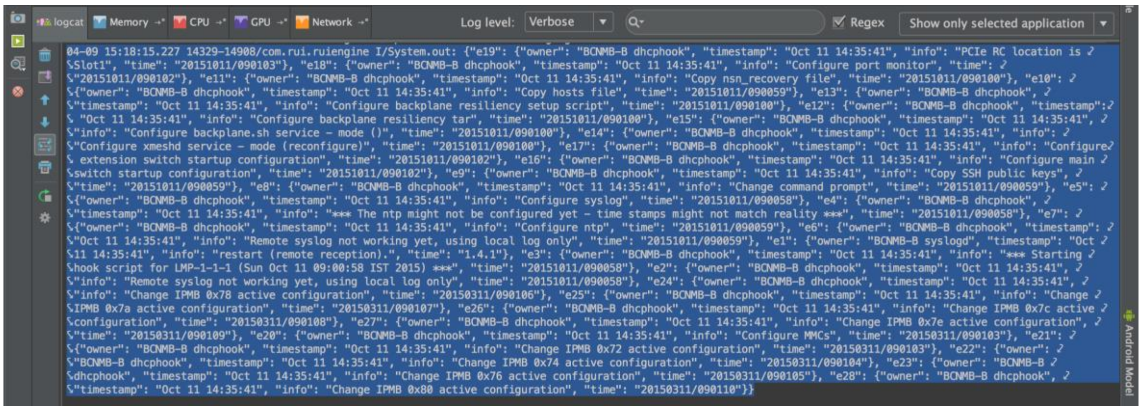
The DisplayBoxSyslogActivity retrieves the system log data from the CPAP, and updates the user interface as below (Figure 32):



```
Oct 11 14:35:41 BCNMB-B syslogd: 1.4.1:
restart (remote reception).
Oct 11 14:35:41 BCNMB-B dhcphook:
Remote syslog working, push local log to
syslog
Oct 11 14:35:41 BCNMB-B dhcphook:
20151011/090058 INFO: Remote syslog not
working yet, using local log only
Oct 11 14:35:41 BCNMB-B dhcphook:
20151011/090058 INFO: *** Starting hook
script for LMP-1-1-1 (Sun Oct 11 09:00:58 IST
2015) ***
Oct 11 14:35:41 BCNMB-B dhcphook:
20151011/090058 INFO: *** The ntp might
not be configured yet - time stamps might not
match reality ***
Oct 11 14:35:41 BCNMB-B dhcphook:
20151011/090058 INFO: Configure syslog
Oct 11 14:35:41 BCNMB-B dhcphook:
20151011/090059 INFO: Remote syslog not
working yet, using local log only
Oct 11 14:35:41 BCNMB-B dhcphook:
20151011/090059 INFO: Configure ntp
Oct 11 14:35:41 BCNMB-B dhcphook:
20151011/090059 INFO: Change command
prompt
Oct 11 14:35:41 BCNMB-B dhcphook:
```

Figure 32. Mobile Client shows system log

If the network is working normally, the system log data is retrieved in the JSON format as shown in Figure 33 below.



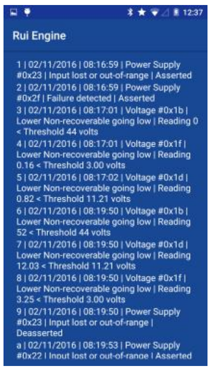
```
04-09 15:18:15.227 14329-14908/com.rui.ruiengine I/System.out: {"e19": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "PCIe RC location is ↲
↳Slot1", "time": "20151011/090103"}, "e18": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Configure port monitor", "time": ↲
↳"20151011/090102"}, "e11": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Copy nsn_recovery file", "time": "20151011/090100"}, "e10": ↲
↳{"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Copy hosts file", "time": "20151011/090059"}, "e13": {"owner": "BCNMB-B dhcphook", ↲
↳"timestamp": "Oct 11 14:35:41", "info": "Configure backplane resiliency setup script", "time": "20151011/090100"}, "e12": {"owner": "BCNMB-B dhcphook", "timestamp":↲
↳ "Oct 11 14:35:41", "info": "Configure backplane resiliency tar", "time": "20151011/090100"}, "e15": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", ↲
↳"info": "Configure backplane.sh service - mode ()", "time": "20151011/090100"}, "e14": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": ↲
↳"Configure xmeshd service - mode (reconfigure)", "time": "20151011/090100"}, "e17": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Configure↲
↳ extension switch startup configuration", "time": "20151011/090102"}, "e16": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Configure main ↲
↳switch startup configuration", "time": "20151011/090102"}, "e9": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Copy SSH public keys", ↲
↳"time": "20151011/090059"}, "e8": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Change command prompt", "time": "20151011/090059"}, "e5": ↲
↳{"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Configure syslog", "time": "20151011/090058"}, "e4": {"owner": "BCNMB-B dhcphook", ↲
↳"timestamp": "Oct 11 14:35:41", "info": "*** The ntp might not be configured yet - time stamps might not match reality ***", "time": "20151011/090058"}, "e7": ↲
↳{"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Configure ntp", "time": "20151011/090059"}, "e6": {"owner": "BCNMB-B dhcphook", "timestamp": ↲
↳"Oct 11 14:35:41", "info": "Remote syslog not working yet, using local log only", "time": "20151011/090059"}, "e1": {"owner": "BCNMB-B syslogd", "timestamp": "Oct ↲
↳11 14:35:41", "info": "restart (remote reception).", "time": "1.4.1"}, "e3": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "*** Starting ↲
↳hook script for LMP-1-1-1 (Sun Oct 11 09:00:58 IST 2015) ***", "time": "20151011/090058"}, "e2": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", ↲
↳"info": "Remote syslog not working yet, using local log only", "time": "20151011/090058"}, "e24": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", ↲
↳"info": "Change IPMB 0x78 active configuration", "time": "20150311/090106"}, "e25": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Change ↲
↳IPMB 0x7a active configuration", "time": "20150311/090107"}, "e26": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Change IPMB 0x7c active ↲
↳configuration", "time": "20150311/090108"}, "e27": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Change IPMB 0x7e active configuration", ↲
↳"time": "20150311/090109"}, "e20": {"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Configure MMCs", "time": "20150311/090103"}, "e21": ↲
↳{"owner": "BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Change IPMB 0x72 active configuration", "time": "20150311/090103"}, "e22": {"owner": ↲
↳"BCNMB-B dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Change IPMB 0x74 active configuration", "time": "20150311/090104"}, "e23": {"owner": "BCNMB-B ↲
↳dhcphook", "timestamp": "Oct 11 14:35:41", "info": "Change IPMB 0x76 active configuration", "time": "20150311/090105"}, "e28": {"owner": "BCNMB-B dhcphook", ↲
↳"timestamp": "Oct 11 14:35:41", "info": "Change IPMB 0x80 active configuration", "time": "20150311/090110"}}
```

Figure 33. System log data in JSON format

The DisplayBoxSelActivity class hierarchy shows as below:

```
public class DisplayBoxSelActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
    private class RetrieveBoxSelsTask extends AsyncTask<Void, Void, String>
    {
        Protected String doInBackground(Void… urls) {}
        Protected void onPostExecute(String s) {}
    }
    public static String HttpRequest(int way, String url) {}
    public void startRetrieveBoxSelsTask() {}
    private List<SelInfo> createList() {}
}
```

The DisplayBoxSelActivity retrieves the system event log data from the CPAP, and updates the user interface as below (Figure 34):



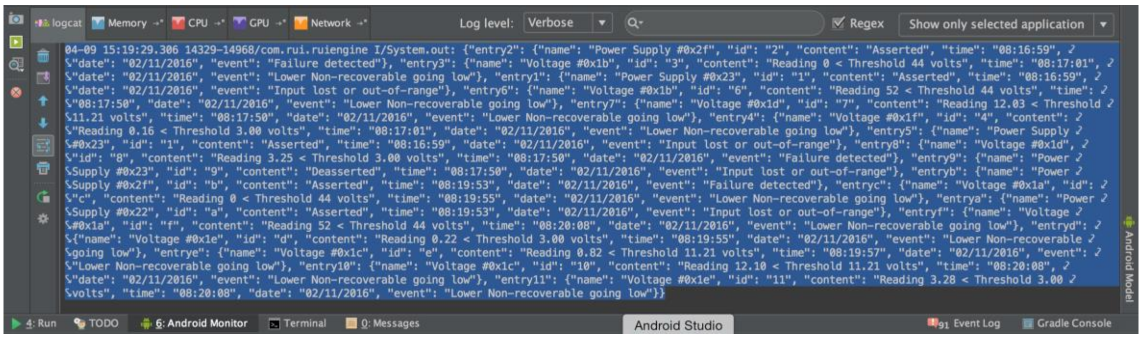
12:37

Rui Engine

1 | 02/11/2016 | 08:16:59 | Power Supply #0x23 | Input lost or out-of-range | Asserted
2 | 02/11/2016 | 08:16:59 | Power Supply #0x2f | Failure detected | Asserted
3 | 02/11/2016 | 08:17:01 | Voltage #0x1b | Lower Non-recoverable going low | Reading 0 < Threshold 44 volts
4 | 02/11/2016 | 08:17:01 | Voltage #0x1f | Lower Non-recoverable going low | Reading 0.16 < Threshold 3.00 volts
5 | 02/11/2016 | 08:17:02 | Voltage #0x1d | Lower Non-recoverable going low | Reading 0.82 < Threshold 11.21 volts
6 | 02/11/2016 | 08:19:50 | Voltage #0x1b | Lower Non-recoverable going low | Reading 52 < Threshold 44 volts
7 | 02/11/2016 | 08:19:50 | Voltage #0x1d | Lower Non-recoverable going low | Reading 12.03 < Threshold 11.21 volts
8 | 02/11/2016 | 08:19:50 | Voltage #0x1f | Lower Non-recoverable going low | Reading 3.25 < Threshold 3.00 volts
9 | 02/11/2016 | 08:19:50 | Power Supply #0x23 | Input lost or out-of-range | Deasserted
a | 02/11/2016 | 08:19:53 | Power Supply #0x22 | Input lost or out-of-range | Asserted

Figure 34. Mobile Client shows system event log

If the network is working normally, the system event log data is retrieved in the JSON format as shown in Figure 35 below.



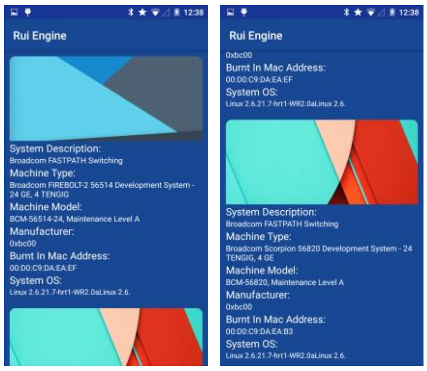
```
04-09 15:19:29.306 14329-14968/com.rui.ruiengine I/System.out: {"entry2": {"name": "Power Supply #0x2f", "id": "2", "content": "Asserted", "time": "08:16:59", ↲
↳"date": "02/11/2016", "event": "Failure detected"}, "entry3": {"name": "Voltage #0x1b", "id": "3", "content": "Reading 0 < Threshold 44 volts", "time": "08:17:01", ↲
↳"date": "02/11/2016", "event": "Lower Non-recoverable going low"}, "entry1": {"name": "Power Supply #0x23", "id": "1", "content": "Asserted", "time": "08:16:59", ↲
↳"date": "02/11/2016", "event": "Input lost or out-of-range"}, "entry6": {"name": "Voltage #0x1b", "id": "6", "content": "Reading 52 < Threshold 44 volts", "time": ↲
↳"08:17:50", "date": "02/11/2016", "event": "Lower Non-recoverable going low"}, "entry7": {"name": "Voltage #0x1d", "id": "7", "content": "Reading 12.03 < Threshold ↲
↳11.21 volts", "time": "08:17:50", "date": "02/11/2016", "event": "Lower Non-recoverable going low"}, "entry4": {"name": "Voltage #0x1f", "id": "4", "content": ↲
↳"Reading 0.16 < Threshold 3.00 volts", "time": "08:17:01", "date": "02/11/2016", "event": "Lower Non-recoverable going low"}, "entry5": {"name": "Power Supply ↲
↳#0x23", "id": "1", "content": "Asserted", "time": "08:16:59", "date": "02/11/2016", "event": "Input lost or out-of-range"}, "entry8": {"name": "Voltage #0x1d", ↲
↳"id": "8", "content": "Reading 3.25 < Threshold 3.00 volts", "time": "08:17:50", "date": "02/11/2016", "event": "Failure detected"}, "entry9": {"name": "Power ↲
↳Supply #0x23", "id": "9", "content": "Deasserted", "time": "08:17:50", "date": "02/11/2016", "event": "Input lost or out-of-range"}, "entryb": {"name": "Power ↲
↳Supply #0x2f", "id": "b", "content": "Asserted", "time": "08:19:53", "date": "02/11/2016", "event": "Failure detected"}, "entryc": {"name": "Voltage #0x1a", "id": ↲
↳"c", "content": "Reading 0 < Threshold 44 volts", "time": "08:19:55", "date": "02/11/2016", "event": "Lower Non-recoverable going low"}, "entrya": {"name": "Power ↲
↳Supply #0x22", "id": "a", "content": "Asserted", "time": "08:19:53", "date": "02/11/2016", "event": "Input lost or out-of-range"}, "entryf": {"name": "Voltage ↲
↳#0x1a", "id": "f", "content": "Reading 52 < Threshold 44 volts", "time": "08:20:08", "date": "02/11/2016", "event": "Lower Non-recoverable going low"}, "entryd": ↲
↳{"name": "Voltage #0x1e", "id": "d", "content": "Reading 0.22 < Threshold 3.00 volts", "time": "08:19:55", "date": "02/11/2016", "event": "Lower Non-recoverable ↲
↳going low"}, "entrye": {"name": "Voltage #0x1c", "id": "e", "content": "Reading 0.82 < Threshold 11.21 volts", "time": "08:19:57", "date": "02/11/2016", "event": ↲
↳"Lower Non-recoverable going low"}, "entry10": {"name": "Voltage #0x1c", "id": "10", "content": "Reading 12.10 < Threshold 11.21 volts", "time": "08:20:08", ↲
↳"date": "02/11/2016", "event": "Lower Non-recoverable going low"}, "entry11": {"name": "Voltage #0x1e", "id": "11", "content": "Reading 3.28 < Threshold 3.00 ↲
↳volts", "time": "08:20:08", "date": "02/11/2016", "event": "Lower Non-recoverable going low"}}
```

Figure 35. System event log data in JSON format

The DisplayBoxFastpathActivity class hierarchy shows as below:

```
public class DisplayBoxFastpathActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
    private class RetrieveBoxFastpathTask extends AsyncTask<Void, Void, String>
    {
        Protected String doInBackground(Void… urls) {}
        Protected void onPostExecute(String s) {}
    }
    public static String HttpRequest(int way, String url) {}
    public void startRetrieveBoxFastpathTask() {}
    private List<FastpathInfo> createList() {}
}
```

The DisplayBoxFastpathActivity retrieves the fastpath data from the CPAP, and updates the user interface as below (Figure 36):



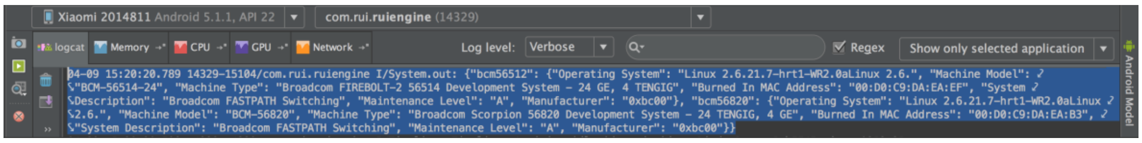
System Description:
Broadcom FASTPATH Switching
Machine Type:
Broadcom FIREBOLT-2 56514 Development System - 24 GE, 4 TENGIG
Machine Model:
BCM-56514-24, Maintenance Level A
Manufacturer:
0xbc00
Burnt In Mac Address:
00:D0:C9:DA:EA:EF
System OS:
Linux 2.6.21.7-hrt1-WR2.0aLinux 2.6.

0xbc00
Burnt In Mac Address:
00:D0:C9:DA:EA:EF
System OS:
Linux 2.6.21.7-hrt1-WR2.0aLinux 2.6.

System Description:
Broadcom FASTPATH Switching
Machine Type:
Broadcom Scorpion 56820 Development System - 24 TENGIG, 4 GE
Machine Model:
BCM-56820, Maintenance Level A
Manufacturer:
0xbc00
Burnt In Mac Address:
00:D0:C9:DA:EA:B3
System OS:
Linux 2.6.21.7-hrt1-WR2.0aLinux 2.6.

Figure 36. Mobile Client shows fastpath

If the network is working normally, the fastpath data is retrieved in the JSON format as shown in Figure 37 below.



Xiaomi 2014811 Android 5.1.1, API 22 | com.rui.ruiengine (14329)

Log level: Verbose | Regex ✓ | Show only selected application

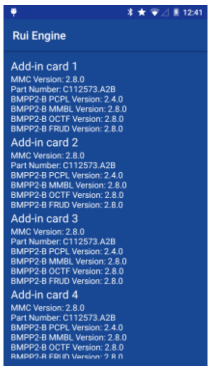
```
04-09 15:20:20.789 14329-15104/com.rui.ruiengine I/System.out: {"bcm56512": {"Operating System": "Linux 2.6.21.7-hrt1-WR2.0aLinux 2.6.", "Machine Model": ↲
↳"BCM-56514-24", "Machine Type": "Broadcom FIREBOLT-2 56514 Development System - 24 GE, 4 TENGIG", "Burned In MAC Address": "00:D0:C9:DA:EA:EF", "System ↲
↳Description": "Broadcom FASTPATH Switching", "Maintenance Level": "A", "Manufacturer": "0xbc00"}, "bcm56820": {"Operating System": "Linux 2.6.21.7-hrt1-WR2.0aLinux ↲
↳2.6.", "Machine Model": "BCM-56820", "Machine Type": "Broadcom Scorpion 56820 Development System - 24 TENGIG, 4 GE", "Burned In MAC Address": "00:D0:C9:DA:EA:B3", ↲
↳"System Description": "Broadcom FASTPATH Switching", "Maintenance Level": "A", "Manufacturer": "0xbc00"}}
```

Figure 37. Fastpath data in JSON format

The DisplayBoxCfgActivity class hierarchy shows as below:

```
public class DisplayBoxCfgActivity extends AppCompatActivity
{
    protected void onCreate(Bundle savedInstanceState) {}
    private class RetrieveBoxCfgTask extends AsyncTask<Void, Void, String>
    {
        Protected String doInBackground(Void… urls) {}
        Protected void onPostExecute(String s) {}
    }
    public static String HttpRequest(int way, String url) {}
    public void startRetrieveBoxCfgTask() {}
    private List<CfgInfo> createList() {}
}
```

The DisplayBoxCfgActivity retrieves the configuration data from the CPAP, and updates the user interface as below (Figure 38):


12:41

Rui Engine

Add-in card 1
MMC Version: 2.8.0
Part Number: C112573.A2B
BMPP2-B PCPL Version: 2.4.0
BMPP2-B MMBL Version: 2.8.0
BMPP2-B OCTF Version: 2.8.0
BMPP2-B FRUD Version: 2.8.0

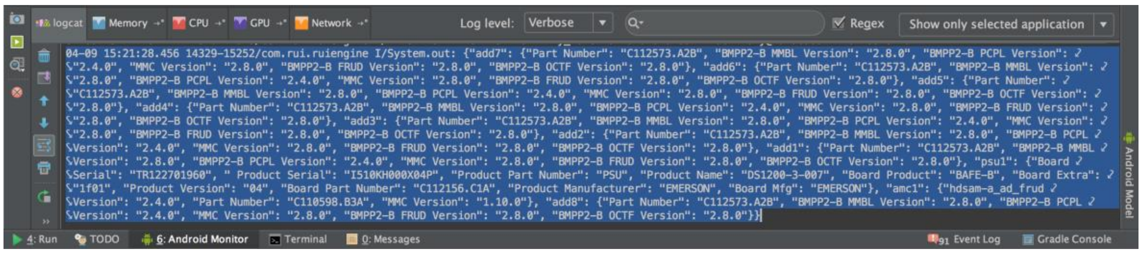
Add-in card 2
MMC Version: 2.8.0
Part Number: C112573.A2B
BMPP2-B PCPL Version: 2.4.0
BMPP2-B MMBL Version: 2.8.0
BMPP2-B OCTF Version: 2.8.0
BMPP2-B FRUD Version: 2.8.0

Add-in card 3
MMC Version: 2.8.0
Part Number: C112573.A2B
BMPP2-B PCPL Version: 2.4.0
BMPP2-B MMBL Version: 2.8.0
BMPP2-B OCTF Version: 2.8.0
BMPP2-B FRUD Version: 2.8.0

Add-in card 4
MMC Version: 2.8.0
Part Number: C112573.A2B
BMPP2-B PCPL Version: 2.4.0
BMPP2-B MMBL Version: 2.8.0
BMPP2-B OCTF Version: 2.8.0
BMPP2-B FRUD Version: 2.8.0

Figure 38. Mobile Client shows configuration

If the network is working normally, the configuration data is retrieved in the JSON format as shown in Figure 39 below.



```
04-09 15:21:28.456 14329-15252/com.rui.ruiengine I/System.out: {"add7": {"Part Number": "C112573.A2B", "BMPP2-B MMBL Version": "2.8.0", "BMPP2-B PCPL Version": ↵
↳"2.4.0", "MMC Version": "2.8.0", "BMPP2-B FRUD Version": "2.8.0", "BMPP2-B OCTF Version": "2.8.0"}, "add6": {"Part Number": "C112573.A2B", "BMPP2-B MMBL Version": ↵
↳"2.8.0", "BMPP2-B PCPL Version": "2.4.0", "MMC Version": "2.8.0", "BMPP2-B FRUD Version": "2.8.0", "BMPP2-B OCTF Version": "2.8.0"}, "add5": {"Part Number": ↵
↳"C112573.A2B", "BMPP2-B MMBL Version": "2.8.0", "BMPP2-B PCPL Version": "2.4.0", "MMC Version": "2.8.0", "BMPP2-B FRUD Version": "2.8.0", "BMPP2-B OCTF Version": ↵
↳"2.8.0"}, "add4": {"Part Number": "C112573.A2B", "BMPP2-B MMBL Version": "2.8.0", "BMPP2-B PCPL Version": "2.4.0", "MMC Version": "2.8.0", "BMPP2-B FRUD Version": ↵
↳"2.8.0", "BMPP2-B OCTF Version": "2.8.0"}, "add3": {"Part Number": "C112573.A2B", "BMPP2-B MMBL Version": "2.8.0", "BMPP2-B PCPL Version": "2.4.0", "MMC Version": ↵
↳"2.8.0", "BMPP2-B FRUD Version": "2.8.0", "BMPP2-B OCTF Version": "2.8.0"}, "add2": {"Part Number": "C112573.A2B", "BMPP2-B MMBL Version": "2.8.0", "BMPP2-B PCPL ↵
↳Version": "2.4.0", "MMC Version": "2.8.0", "BMPP2-B FRUD Version": "2.8.0", "BMPP2-B OCTF Version": "2.8.0"}, "add1": {"Part Number": "C112573.A2B", "BMPP2-B MMBL ↵
↳Version": "2.8.0", "BMPP2-B PCPL Version": "2.4.0", "MMC Version": "2.8.0", "BMPP2-B FRUD Version": "2.8.0", "BMPP2-B OCTF Version": "2.8.0"}, "psu1": {"Board ↵
↳Serial": "TR122701960", " Product Serial": "I510KH000X04P", "Product Part Number": "PSU", "Product Name": "DS1200-3-007", "Board Product": "BAFE-B", "Board Extra": ↵
↳"1f01", "Product Version": "04", "Board Part Number": "C112156.C1A", "Product Manufacturer": "EMERSON", "Board Mfg": "EMERSON"}, "amc1": {"hdsam-a_ad_frud ↵
↳Version": "2.4.0", "Part Number": "C110598.B3A", "MMC Version": "1.10.0"}, "add8": {"Part Number": "C112573.A2B", "BMPP2-B MMBL Version": "2.8.0", "BMPP2-B PCPL ↵
↳Version": "2.4.0", "MMC Version": "2.8.0", "BMPP2-B FRUD Version": "2.8.0", "BMPP2-B OCTF Version": "2.8.0"}}
```

Figure 39. Configuration data in JSON format

With the four main function units: Initialization, Access Control, Data Management, and Analytics implemented, the MC meets the requirement of the system architecture on the mobile client side. In the next section, the implementation of the CPAP will be described.

4.2 Cloud Platform API Provider

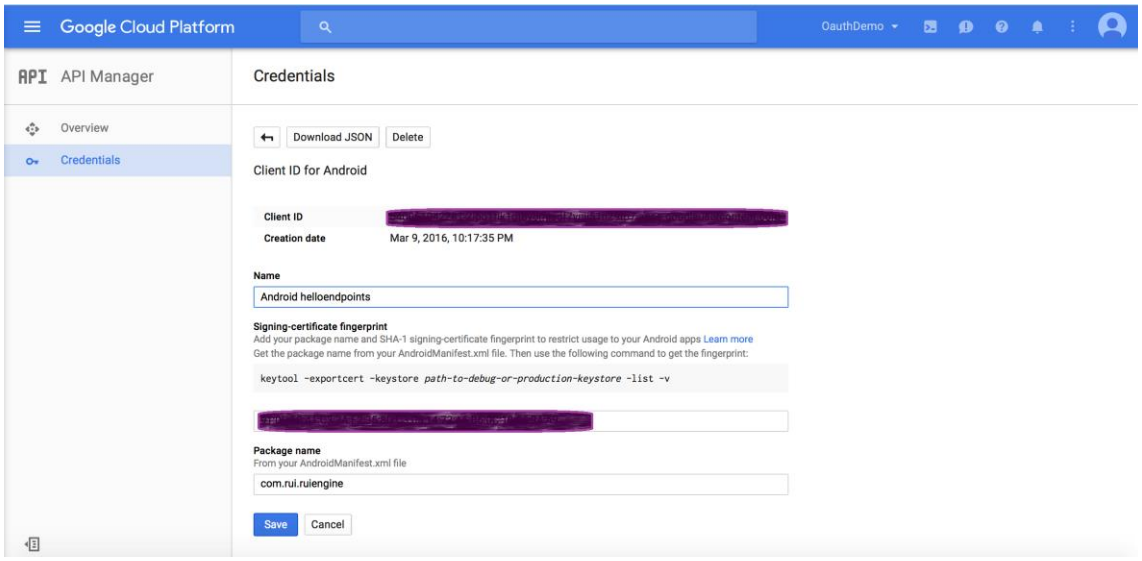
The CPAP implemented for this project is based on GAE. It has two main separate parts: the service authenticator and the content provider. Both the service authenticator and the content provider are written in the Python programming language.

For the service authenticator, it implements Oauth2.0 authentication server by using Google Cloud Endpoints solution directly. After the service authenticator project deployment to the Google App Engine, it is required to register the Android Client that developed in this project so that it can access to the Google Cloud Endpoints. To make it happen, these steps are followed in the Google Cloud Platform console:

- Navigate to the Google Cloud Console and go into API Manager section
- Select Credentials menu, and create the Client ID for Android application
- Fill in all must content

  Get Android application debugging keystore fingerprint by “keytool -list -v -keystore ~/.android/debug.keystore”
- Provide the Android Application package name

After these are completed, the Client ID is created. Figure 40 below shows the client ID registration in the Google Cloud Platform.



Credentials

Download JSON | Delete

Client ID for Android

Client ID [redacted]
Creation date Mar 9, 2016, 10:17:35 PM

Name
Android helloendpoints

Signing-certificate fingerprint
Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps Learn more
Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:

```
keytool -exportcert -keystore path-to-debug-or-production-keystore -list -v
```

[redacted]

Package name
From your AndroidManifest.xml file
com.rui.ruiengine

Save | Cancel

Figure 40. Client ID registration in Google Cloud Platform

The registered Client ID for the authenticator allows the MC application get itself to be authenticated, that means another Android application is not permitted to get the authentication service.

For the content provider, it is webapp2 based application designed in the GAE. The main feature of the content providers provide the needed raw data to the Android application client when it receives the request.

5 Discussion and Conclusions

The project involves the practical work of designing and developing the mobile client application and the cloud platform service. The work is dedicated for a traditional telecom enterprise, in order to introduce the mobile concept for the daily R&D maintenance work.

The main technical challenge for the project was the complexity of the whole system architecture design, it has both front-end application and back-end service. Hence, different technologies including mobile computing, cloud computing, web service are integrated for the system. With several rounds of investigation of the most popular and powerful mobile technology, the project fixes the system architecture eventually. The mobile client as the front-end application is implemented based on Android platform, while the cloud platform as the back-end service is implemented based on Google App Engine. Considering the implementation work, there is a constraint when demanding real data from the company's secure network. It is forbidden to access the laboratory network without the company's IT grant. To solve the access issue, the project chose the solution of simulating the real equipment data. The simulation is performed in the cloud platform. It is assumed the simulated data can reflect the real data transmission between the DS and the CPAP. The method of simulation is out of the project scope.

From the business challenge point of view, the project is to convince that the mobile concept can be feasible and powerful for the future work in the telecom enterprise. Binding the traditional telecom equipment industry with new mobile concept needs appropriate entry points. In the telecom enterprise, the efficiency of daily R&D maintenance work usually determines an important Key Performance Indicator (KPI) performance. It is a demanding to introduce the new way of maintenance work. The maintenance tool is the most important factor. With common understanding from the company, the project took the daily R&D maintenance work as the good entry point.

In the project, the ESA system architecture was designed, the ESA system implementation was partially completed, and there are two parts: the CPAP and light-weight MC. The outcome of the project provides the evidence of the feasibility of applying the mobile application in the telecom enterprise work. The ESA system prototype was developed and delivered internally with some limitations listed here:

- The access control to the MC relies on the Google account, for the telecom enterprise, it is not secure and safe
- There is no multi user support in the system
- On the CPAP side, the data source is not really generated by the DUT, to just prove the concept, the data is simulated and generated locally in the CPAP
- There is no data persistence implementation on both the MC and CPAP side. All data existing in the system are volatile.
- The heavy analytics part is not yet powerfully developed, hence, the analysis result shown on the MC is not that smart as it is expected to be.

All above mentioned limitations or incompleteness will be continued to be worked on in future. The plan as follows:

- The access control implemented in both MC and CPAP relies on the company's own authentication policy and intra-account
- Must support multiple users in the system. This is quite important to improve the efficiency of the maintenance cooperation work
- Getting the access to the real equipment, and retrieve the real data
- Implementing data store in the CPAP
- Improving analytics feature in the CPAP.

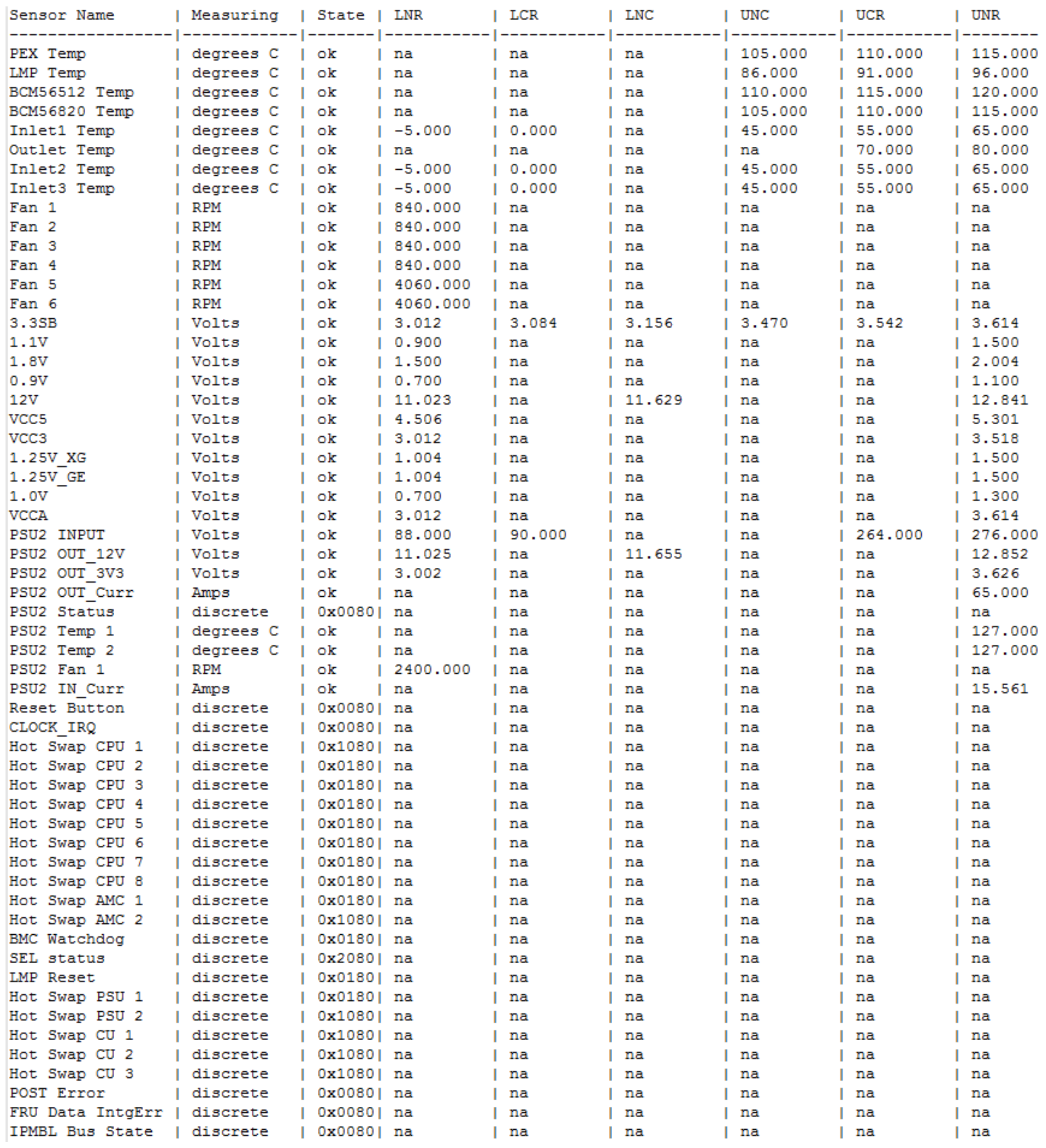
Although there are limitations or incompleteness in the project, for the academic purpose, the project provides the valuable practice of introducing the mobile concept to the conservative telecom industry, and in this direction, the mobile concept has a bright feature.

References

1. Wikipedia, Mobile app. [ONLINE]. Available: https://en.wikipedia.org/wiki/Mobile_app. [Accessed December 12, 2015]
2. Renee Bassett, Industrial Mobile Apps: Who's Using Them and Why. August 29, 2014. [ONLINE]. Available: http://www.automationworld.com/mobility/industrial-mobile-apps-whos-using-them-and-why. [Accessed December 12, 2015]
3. Tomi T Ahonen, Smartphone Wars: Q3 Scorecard - All market shares, Top 10 brands, OS platforms, Installed base. October 30, 2015. [ONLINE]. Available: http://communities-dominate.blogs.com/brands/2015/10/smartphone-wars-q3-scorecard-all-market-shares-top-10-brands-os-platforms-installed-base.html. [Accessed December 15, 2015]
4. Google, Android Studio Overview. [ONLINE]. Available: http://developer.android.com/tools/studio/index.html. [Accessed December 15, 2015]
5. Google, Android Interfaces and Architecture. [ONLINE]. Available: https://source.android.com/devices/. [Accessed December 18, 2015]
6. Wikipedia, Android version history. [ONLINE]. Available: https://en.wikipedia.org/wiki/Android_version_history. [Accessed December 26, 2015]
7. Google, App Engine. [ONLINE]. Available: https://cloud.google.com/appengine/. [Accessed December 14, 2015]
8. Sanderson, Dan (2009). Programming Google App Engine: Build and Run Scalable Web Apps on Google's Infrastructure. O'Reilly Media.ISBN 978-0-596-52272-8. p1 - 3.
9. Google, Overview of App Engine Features. [ONLINE]. Available: https://cloud.google.com/appengine/features/. [Accessed December 29, 2015]
10. Wikipedia, Hypertext Transfer Protocol. [ONLINE]. Available: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Accessed January 4, 2016]
11. Charles M. Kozierok, HTTP Operational Model and Client /Server Communication. September 20, 2005. [ONLINE]. Available: http://www.tcpipguide.com/free/t_HTTPOperationalModelandClientServerCommunication.htm. [Accessed January 4, 2016]
12. Charles M. Kozierok, HTTP Request Message Format. September 20, 2005. [ONLINE]. Available: http://www.tcpipguide.com/free/t_HTTPRequestMessageFormat.htm. [Accessed January 5, 2016]

13. Charles M. Kozierok, HTTP Response Message Format. September 20, 2005. [ONLINE]. Available: http://www.tcpipguide.com/free/t_HTTPResponseMessageFormat.htm. [Accessed January 6, 2016]
14. Wikipedia, Secure Shell. [ONLINE]. Available: https://en.wikipedia.org/wiki/Secure_Shell. [Accessed January 10, 2016]
15. VanDyke Software, Inc, White Paper, An Overview of the Secure Shell (SSH). p3 - 6.
16. Wikipedia, Oauth. [ONLINE]. Available: https://en.wikipedia.org/wiki/OAuth. [Accessed January 20, 2016]
17. Mohammad Amin Saghizadeh, INTRODUCING OAUTH 2.0 PROTOCOL: SECURITY AND PERFORMANCE. [ONLINE]. Available: http://www.myjittips.com/introducing-oauth-2-security-and-performance/. [Accessed January 21, 2016]
18. Google, Mobile App Backend Services. [ONLINE]. Available: https://cloud.google.com/solutions/mobile/mobile-app-backend-on-cloud-platform. [Accessed February 2, 2016]

DUT Sensors Table



```
Sensor Name      | Measuring  | State | LNR       | LCR       | LNC       | UNC       | UCR       | UNR
-----------------|------------|-------|-----------|-----------|-----------|-----------|-----------|--------
PEX Temp         | degrees C  | ok    | na        | na        | na        | 105.000   | 110.000   | 115.000
LMP Temp         | degrees C  | ok    | na        | na        | na        | 86.000    | 91.000    | 96.000
BCM56512 Temp    | degrees C  | ok    | na        | na        | na        | 110.000   | 115.000   | 120.000
BCM56820 Temp    | degrees C  | ok    | na        | na        | na        | 105.000   | 110.000   | 115.000
Inlet1 Temp      | degrees C  | ok    | -5.000    | 0.000     | na        | 45.000    | 55.000    | 65.000
Outlet Temp      | degrees C  | ok    | na        | na        | na        | na        | 70.000    | 80.000
Inlet2 Temp      | degrees C  | ok    | -5.000    | 0.000     | na        | 45.000    | 55.000    | 65.000
Inlet3 Temp      | degrees C  | ok    | -5.000    | 0.000     | na        | 45.000    | 55.000    | 65.000
Fan 1            | RPM        | ok    | 840.000   | na        | na        | na        | na        | na
Fan 2            | RPM        | ok    | 840.000   | na        | na        | na        | na        | na
Fan 3            | RPM        | ok    | 840.000   | na        | na        | na        | na        | na
Fan 4            | RPM        | ok    | 840.000   | na        | na        | na        | na        | na
Fan 5            | RPM        | ok    | 4060.000  | na        | na        | na        | na        | na
Fan 6            | RPM        | ok    | 4060.000  | na        | na        | na        | na        | na
3.3SB            | Volts      | ok    | 3.012     | 3.084     | 3.156     | 3.470     | 3.542     | 3.614
1.1V             | Volts      | ok    | 0.900     | na        | na        | na        | na        | 1.500
1.8V             | Volts      | ok    | 1.500     | na        | na        | na        | na        | 2.004
0.9V             | Volts      | ok    | 0.700     | na        | na        | na        | na        | 1.100
12V              | Volts      | ok    | 11.023    | na        | 11.629    | na        | na        | 12.841
VCC5             | Volts      | ok    | 4.506     | na        | na        | na        | na        | 5.301
VCC3             | Volts      | ok    | 3.012     | na        | na        | na        | na        | 3.518
1.25V_XG         | Volts      | ok    | 1.004     | na        | na        | na        | na        | 1.500
1.25V_GE         | Volts      | ok    | 1.004     | na        | na        | na        | na        | 1.500
1.0V             | Volts      | ok    | 0.700     | na        | na        | na        | na        | 1.300
VCCA             | Volts      | ok    | 3.012     | na        | na        | na        | na        | 3.614
PSU2 INPUT       | Volts      | ok    | 88.000    | 90.000    | na        | na        | 264.000   | 276.000
PSU2 OUT_12V     | Volts      | ok    | 11.025    | na        | 11.655    | na        | na        | 12.852
PSU2 OUT_3V3     | Volts      | ok    | 3.002     | na        | na        | na        | na        | 3.626
PSU2 OUT_Curr    | Amps       | ok    | na        | na        | na        | na        | na        | 65.000
PSU2 Status      | discrete   | 0x0080| na        | na        | na        | na        | na        | na
PSU2 Temp 1      | degrees C  | ok    | na        | na        | na        | na        | na        | 127.000
PSU2 Temp 2      | degrees C  | ok    | na        | na        | na        | na        | na        | 127.000
PSU2 Fan 1       | RPM        | ok    | 2400.000  | na        | na        | na        | na        | na
PSU2 IN_Curr     | Amps       | ok    | na        | na        | na        | na        | na        | 15.561
Reset Button     | discrete   | 0x0080| na        | na        | na        | na        | na        | na
CLOCK_IRQ        | discrete   | 0x0080| na        | na        | na        | na        | na        | na
Hot Swap CPU 1   | discrete   | 0x1080| na        | na        | na        | na        | na        | na
Hot Swap CPU 2   | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap CPU 3   | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap CPU 4   | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap CPU 5   | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap CPU 6   | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap CPU 7   | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap CPU 8   | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap AMC 1   | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap AMC 2   | discrete   | 0x1080| na        | na        | na        | na        | na        | na
BMC Watchdog     | discrete   | 0x0180| na        | na        | na        | na        | na        | na
SEL status       | discrete   | 0x2080| na        | na        | na        | na        | na        | na
LMP Reset        | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap PSU 1   | discrete   | 0x0180| na        | na        | na        | na        | na        | na
Hot Swap PSU 2   | discrete   | 0x1080| na        | na        | na        | na        | na        | na
Hot Swap CU 1    | discrete   | 0x1080| na        | na        | na        | na        | na        | na
Hot Swap CU 2    | discrete   | 0x1080| na        | na        | na        | na        | na        | na
Hot Swap CU 3    | discrete   | 0x1080| na        | na        | na        | na        | na        | na
POST Error       | discrete   | 0x0080| na        | na        | na        | na        | na        | na
FRU Data IntgErr | discrete   | 0x0080| na        | na        | na        | na        | na        | na
IPMBL Bus State  | discrete   | 0x0080| na        | na        | na        | na        | na        | na
```

Credit

All icons used in the mobile applications are made by Freepik from www.flati-
con.com:

<div>Icons made by <a href="http://www.freepik.com" title="Freepik">Freepik</a> from <a
href="http://www.flaticon.com" title="Flaticon">www.flaticon.com</a> is licensed by <a
href="http://creativecommons.org/licenses/by/3.0/" title="Creative Commons BY 3.0" tar-
get="_blank">CC 3.0 BY</a></div>