Wi-Fi subsystem of RDK-B stack

☒ **Draft**
□ Baselined for Estimates

Author	
Authorized by	
Version	0.1
Date	18-APR-2014

CONFIDENTIALITY STATEMENT
The information contained herein is proprietary to Comcast and may not be used, reproduced or disclosed to others except as specifically permitted in writing by Comcast. The recipient of this document, by its retention and use, agrees to protect the information contained herein.

Table of Contents

List of Tables

1. INTRODUCTION

This document covers the architecture details of Wi-Fi subsystem which is part of the RDK-B code base used for XB3. Description includes the Intel ATOM specific implementation details followed in the architecture.

1.1 Document Scope and Objectives

Scope of this document is to explain the Wi-Fi subsystem of RDK-B stack.

1.2 Revision History

This revision history only documents versions that are distributed to multiple team members.

Version	Release Date	Description
0.1	18-APR-2014	Initial draft version

Table 1 Revision History

1.3 References

#	Name	Owner/Author	Version	Date
1	CCSP Common Component Software Architecture Specification			
2				

Table 2 Reference Documents

1.4 Legend

#	Notation	Description
1		
2		

Table 3 Legend

1.5 Stakeholder and Review Approval / Sign off

Group	Name	Role

Table 4 Stakeholder List

1.6 Definitions and Acronyms

ACRONYMS

Acronym	Description

Table 5 Acronyms

DEFINITIONS

Term	Description

Table 6 Definitions

1.7 Assumptions and Risks

1.8 Dependencies on other projects / features

2. Architecture of Wi-Fi subsystem

Wi-Fi sub-system is part of Provisioning and Management (P&M) module of gateway. Sub-system supports all the parameters, which are defined in the TR-181 data-model.
Figure 1 shows the architecture diagram of P&M module in the gateway stack. Here, Access Layer takes care of call to/from D-Bus. It provides all API's for get/set parameters based on request from various other modules in the gateway.

Data-model Management Layer (DML) loads all data model access APIs through a pre-defined XML file TR181-WiFi-USGV2.XML. Whole data-model implementation is through a shared library libwifi.so.

DML interacts with the hardware/WiFi driver through an abstraction layer which converts the user space calls to system level calls and takes the necessary action at driver.

HAL Integration Layer provides necessary APIs to upper and lower layers to communicate. HAL has a plugin layer, which will interact with the common data-model libraries and facilitates the component specific management.
Plug-in module of HAL will then call the back end manager so as to initialize and manage Wi-Fi components.
Later, HAL will interact with the driver specific APIs to configure Wi-Fi.

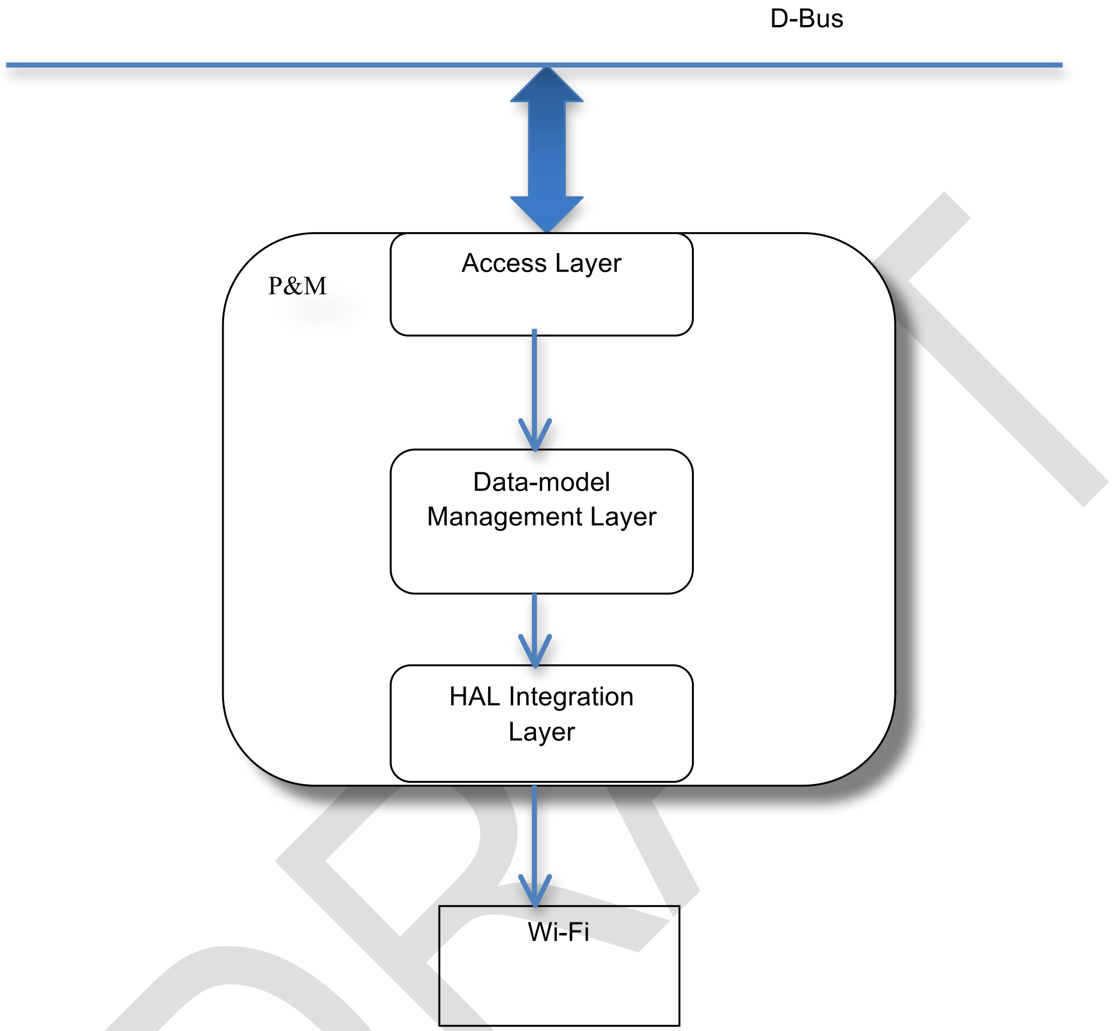


Figure 1 Architectural view of Wi-Fi sub-system

3. Initialization sequence of Wi-Fi

Once the Wi-Fi driver comes up at platform layer there are a sequence of operations carried at different layers of Wi-Fi subsystem of the gateway. Wi-Fi initialization is carried out by different layers, which are part of P&M module of the home gateway.

During gateway boot-up, the XML file which has all information of the objects, parameters and the APIs will be registered to the common component library. In case of Wi-Fi the XML file used to populate the object related information is TR181-WiFi-USGV2.XML. Once the component specific library is loaded, the APIs are registered and then the component specific initialization happens.

During the process of component specific data-model initialization when P&M gets the component specific plugin initialization command from CCSP data-model library, corresponding back end manager is called to initialize the Wi-Fi. Procedure flow of Wi-Fi initialization is depicted in Figure 2 below.

Wi-Fi driver initialization is carried by the component specific API layer, which will try to fetch the parameters from PSM and apply them to the driver if needed and reinitialize the Access Points. As shown in Figure 2, Wi-Fi P&M follows the layered approach. Middle layer provides the necessary APIs to the common component so as to interact with the component specific layer. Middle layer then interact with the Integration layer whose responsibility is to mainly deal with the data-model APIs, driver specific APIs or dbus APIs to manage Wi-Fi specific parameters. Integration layer will act as a gluing layer between the driver/dbus/P&M databases and the upper layers.

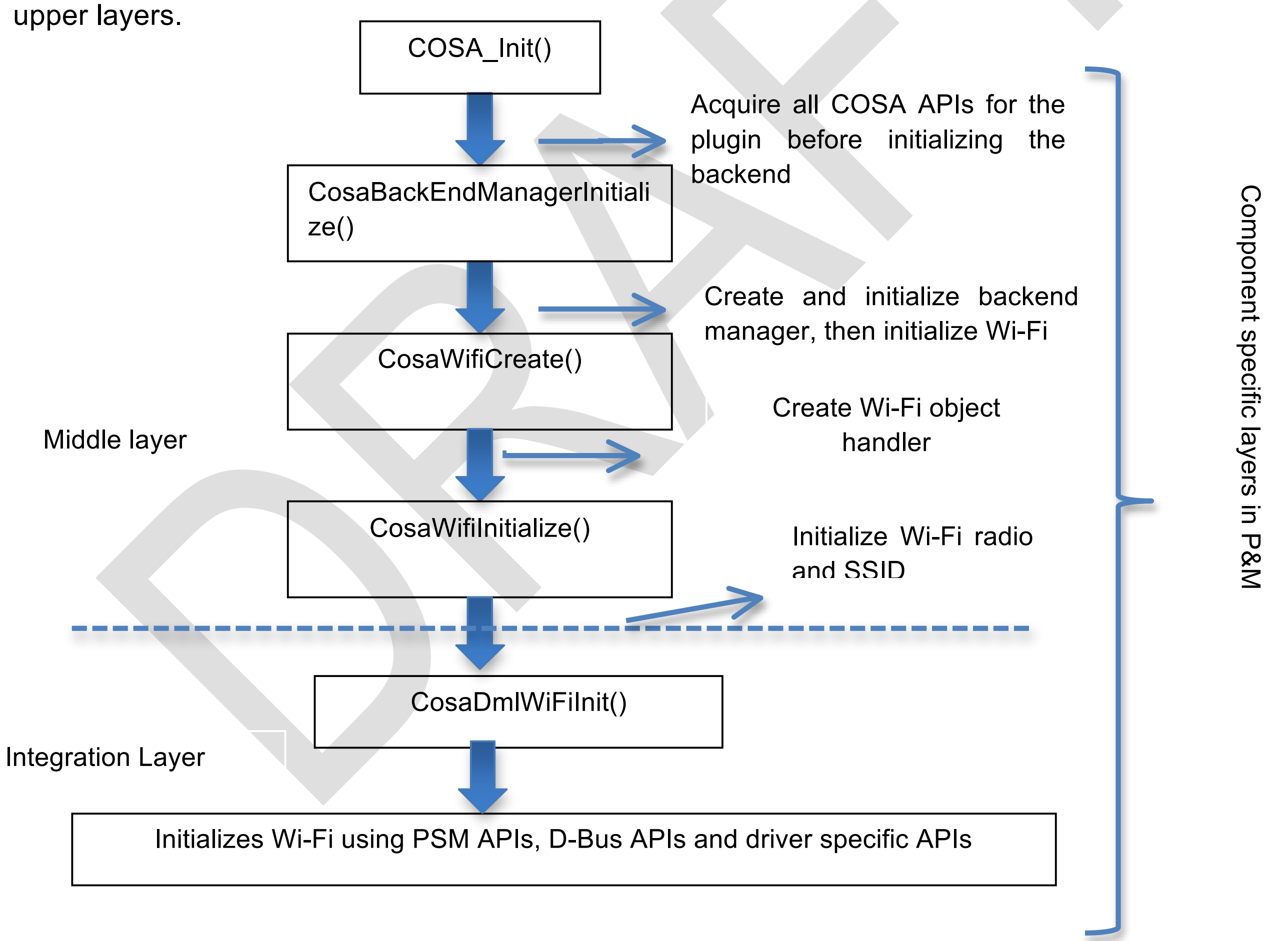

COSA_Init()
Acquire all COSA APIs for the plugin before initializing the backend
CosaBackEndManagerInitialize()
Create and initialize backend manager, then initialize Wi-Fi
CosaWifiCreate()
Create Wi-Fi object handler
Middle layer
CosaWifiInitialize()
Initialize Wi-Fi radio and SSID
CosaDmlWiFiInit()
Integration Layer
Initializes Wi-Fi using PSM APIs, D-Bus APIs and driver specific APIs
Component specific layers in P&M

Figure 2 Wi-Fi initialization

4. Code flow for getting a Wi-Fi parameter

During initialization phase of gateway, all the necessary APIs related to an object mentioned in the data-model USG file will be registered with the data-model management system. So if a query comes for a particular parameter of an object, the registered API for that particular parameter type will be called so that the information can be retrieved from the P&M module.

Once the query comes from UI/SNMP/TR69, the PA accesses corresponding D-BUS API. Later the object is identified from the registry and the corresponding API call specific to the parameter being queried is accessed. API to be accessed is determined by the data-model XML which is used for the library. Then the parameter retrieval process end up in retrieving the value for the parameter from a centralized database maintained by P&M sub-system.

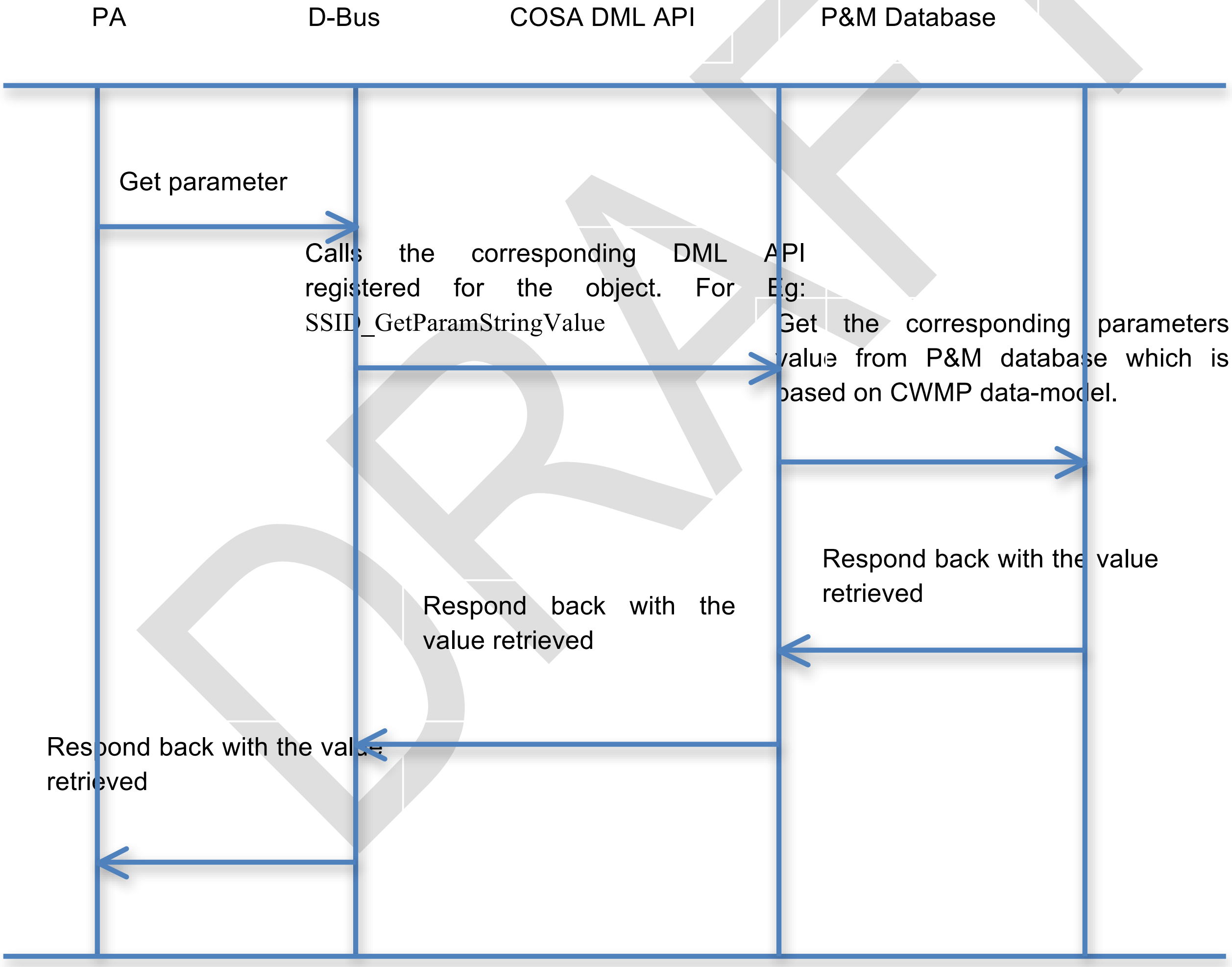

PA
D-Bus
COSA DML API
P&M Database
Get parameter
Calls the corresponding DML API registered for the object. For Eg: SSID_GetParamStringValue
Get the corresponding parameters value from P&M database which is based on CWMP data-model.
Respond back with the value retrieved
Respond back with the value retrieved
Respond back with the value retrieved

Figure 3 : Code flow of Wi-Fi subsystem

5. Code details of RDK-B implementations

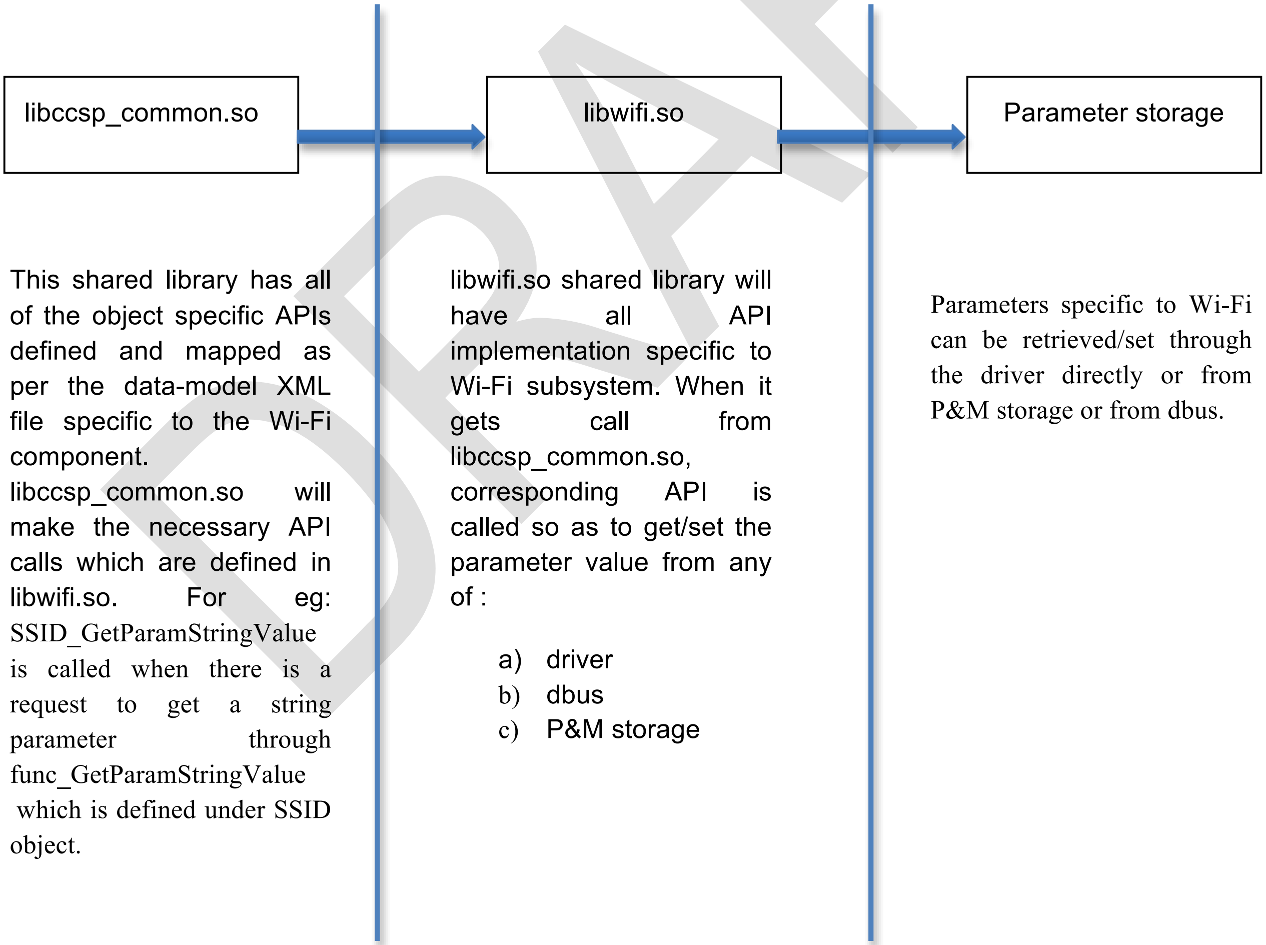
Wi-Fi subsystem is made as part of a shared library libwifi.so in ATOM platform.

5.1 Intel ATOM specific HAL

All Intel specific code is embraced using a compilation flag _COSA_INTEL_USG_ATOM_.

For the current implementation used for XB3 where Wi-Fi driver runs on ATOM platform, all driver specific APIs which are called for setting AP and radio specific parameters are through athcfg_api.c. Component Specific APIs defined in RDK-B architecture will be interacting with the driver through these APIs.

Interactions of Wi-Fi libraries are as below:



This shared library has all of the object specific APIs defined and mapped as per the data-model XML file specific to the Wi-Fi component. libccsp_common.so will make the necessary API calls which are defined in libwifi.so. For eg: SSID_GetParamStringValue is called when there is a request to get a string parameter through func_GetParamStringValue which is defined under SSID object.

libwifi.so shared library will have all API implementation specific to Wi-Fi subsystem. When it gets call from libccsp_common.so, corresponding API is called so as to get/set the parameter value from any of :

a) driver
b) dbus
c) P&M storage

Parameters specific to Wi-Fi can be retrieved/set through the driver directly or from P&M storage or from dbus.

Figure 4 : Library interactions involved

5.2 RDK-B HAL

All Wi-Fi related API's are even present as part of Utopia which is for the time being is not used in the code and is commented out from the build scripts.