**CONTENTS :**

* **B2g OS Overall Architecture ........... 02**
* **Gecko Architecture ........... 05**
* **Gonk Architecture ........... 07**
* **Steps for Building code ........... 09**
* **Emulator ........... 10**
* **References ........... 11**

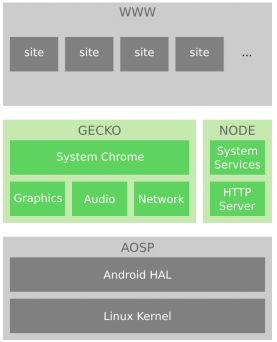
**B2G OS**

**B2G ( Short for Boot to Gecko)**

**B2G OS** : B2G OS is an operating system for smart phones manintained by Mozilla community.

B2G OS eliminates the native API layer between the operating system and application layers. This integrated design reduces platform overhead and simplifies security without sacrificing performance or a rich user smart phone experience.

**Overall Architecture :**



**Gaia** is the core web apps of the device, and user interface layer, all written in HTML5, CSS and JavaScript, with a number of exposed APIs to allow the UI code to interact with the phone hardware and Gecko functionality.

The gaia is in the following location

**B2g/gaia/apps**

**B2g/gaia/webapps**

**Gecko** is the web engine and presentation layer in B2G OS that connects hardware to HTML by serving as the interface between web content and the underlying device. Gecko provides an HTML5 parsing and rendering engine, programmatic access to hardware functionality via secure web APIs, a comprehensive security framework, update management, and other core services.

The gecko is in the following location

**B2g/gecko/**

**Gonk** is the kernel-level component in the B2G OS stack that serves as the interface between Gecko and the underlying hardware. Gonk controls the underlying hardware and exposes hardware capabilities to Web APIs implemented in Gecko. Gonk can be seen as the “black box” that does all the complex, detailed work behind the scenes to control the mobile device by enacting requests at the hardware level.

**The Linux Kernel** will bring up devices and run essential processes. It will execute processes defined in init.rc and the successor init.b2g.rc to boot essential process such as b2g (FirefoxOS basic process, containing Gecko) and rild (telephony related process that might proprietary by different chipsets).At the end of the process, a userspace init process is launched.

* Once the init process is launched, the Linux kernel handles system calls from userspace, and interrupts and the like from hardware devices. Many hardware features are exposed to userspace through sysfs.

There is a pre built kernel present which is used while building emulator.

**B2G/prebuilts/qemu-kernel**

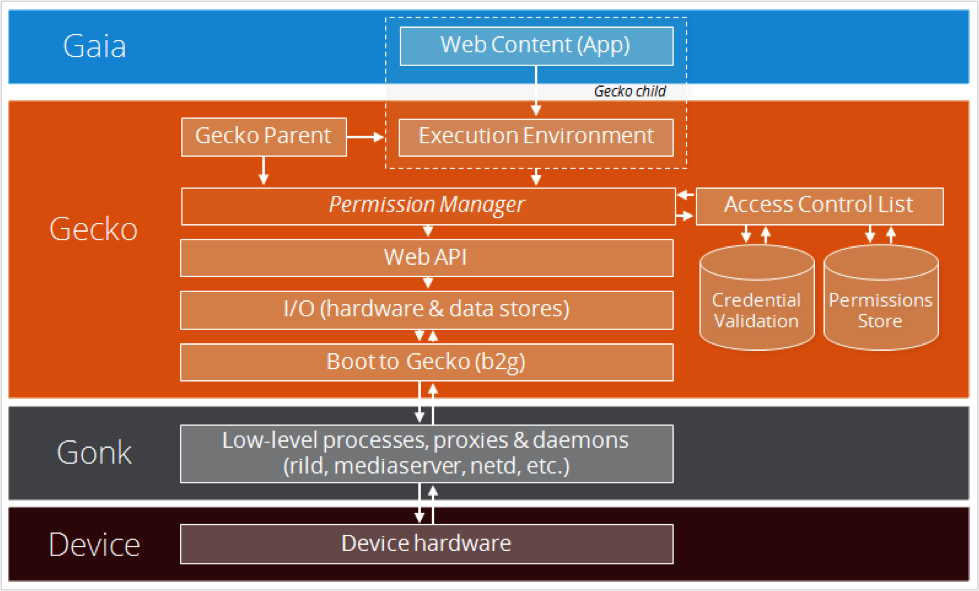
**Hardware Abstraction Layer (HAL)**

The Gecko hardware abstraction layer is one of the porting layers of Gecko. It handles low-level access to system interfaces across multiple platforms using a C++ API that's accessible to the higher levels of Gecko. These APIs are implemented on a per-platform basis inside the Gecko HAL itself. This hardware abstraction layer is not exposed directly to JavaScript code in Gecko — this part of the interaction is handled by the Web APIs.

The HAL is in the following location

**B2G/gecko/hal**

**Gecko architecture diagram :**



* **Security Framework:** contains

**Permission Manager:** Gateway to accessing functionality in the Web API.

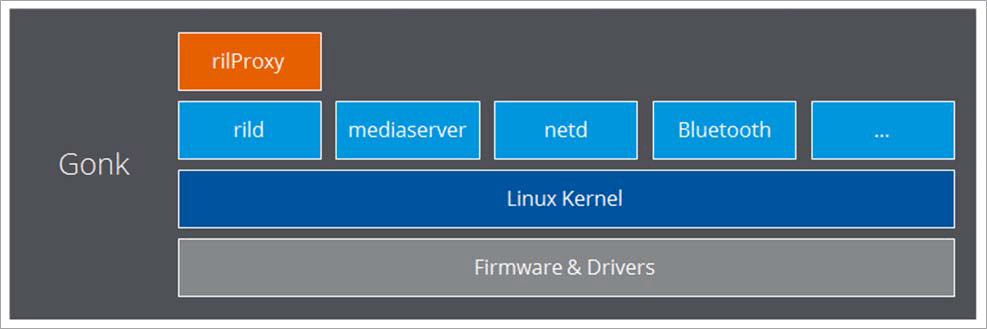
**Access Control List:** Matrix of roles and permissions required to access Web API functionality.

**Credential Validation:** Authentication of apps/users.

**Permissions Store:** Set of privileges required to access Web API functionality.

* **Web API:** Set of standard APIs that exposes hardware functionality to web content. Provides web apps with secure, programmatic access to features in the underlying mobile device hardware, along with data that is stored on or available to a device.
* **I/O:** Interface to the hardware and data store.
* **Software Updates:** Obtain and install updates to system software and third-party apps.
* **Content Layout & Rendering:** Engine that parses, interprets, and executes web content and, with formatting information, displays the formatted content to the user.
* **b2g process:** (Gecko) runs in a highly-privileged system process that has access to hardware features in the mobile phone. Running apps are child processes of b2g.

**Gonk architecture diagram :**

****

* Gonk is the lower level operating system of the B2G OS platform, consisting of a Linux kernel based on the Android Open Source Project (AOSP) and userspace hardware abstraction layer (HAL).
* Gonk is a device porting layer: an adapter between the hardware and Gecko. Gonk is a relatively simple Linux distribution that can be treated as a Gecko Port paired with Gecko porting layers — so Gonk is a porting target of Gecko, just like there's a port of Gecko to OS X, Windows, and Android.
* Each mobile phone model has a specialized combination of Gonk components based on the system libraries, device drivers, and firmware required to operate the device. These components are determined by the OEM in collaboration with chipset manufacturers and ODMs. The above figure shows an example Gonk implementation:

This example shows the following main components (which represent only a subset of the possible components in any given Gonk implementation):

**Linux Kernel:** Uses libraries from Android (GPS, camera, etc.) and other open source projects (Linux, libusb, bluez, and so on).

**Radio Interface Layer (RIL):** Interacts with the modem hardware (telephony) in the phone. Consists of two components:

**rild daemon:** Talks to the modem firmware.

**rilProxy:** Proxies messages between rild and the b2g process

**mediaserver process:** Controls audio and video playback. Gecko communicates with the media server through an Android RPC mechanism.

**netd process:** Network daemon that interacts directly with network interfaces (Wi-fi) in the hardware.

**Bluetooth, etc.:** Bluetooth and other service-level daemons provide access to hardware capabilities.

**Steps for building b2g Emulator :**

**1.** Give the following commands

git clone git://github.com/mozilla-b2g/B2G.git

cd B2G

./config.sh emulator-l

**2.** Replace or merge changes from attached makefile into B2G/gaia/

**3.** Download android-ndk-r11c from the https://dl.google.com/android/repository/android-ndk-r20b-linux-x86\_64.zip and put it in your /home/username/.mozbuild/ folder and rename it to android-ndk-r11b. Ex:- /home/username/.mozbuild/android-ndk-r11b

If .mozbuild is not there , create it with same name

**4.** Move all the manifests xml file from .repo/manifests/ to .repo

**5.** Delete out folder, any sdk under B2G/gaia/b2g\_sdk as wrong file got downloaded earlier, objdir-gecko.

**6.** Run ./build.sh command to build the code .

**7.** Run ./run-emulator.sh

**What is Emulator ?**

An Emulator is a piece of code that allows you to run ANOTHER software (usually a whole OS) on the hosting Operation system. The Emulator shares the same hardware as it’s host. (Thus the recommendation not to open other apps while using an emulator).

So start the B2G emulator, type the following command:

**./run-emulator.sh**

This will handle all the emulator startup tasks for you. Now wait patiently while the emulator starts up and Boot to Gecko boots up on it.

**When the emulator doesn't work :**

Sometimes the emulator fails to start up. Here are some tips for resolving problems..

* Make sure ADB server is running
* Erase Configuration settings.

**Facing Issue :**

* **Blank screen issue**

**REFERENCES** :

<https://developer.mozilla.org/en-US/docs/Archive/B2G_OS/Porting_B2G_OS/basics>

<https://developer.mozilla.org/en-US/docs/Archive/B2G_OS/Architecture>

<https://developer.mozilla.org/en-US/docs/Archive/B2G_OS/Using_the_B2G_emulators>