FAQ-RDK-B

- The rootfs doesn't contain libpcap.so, as tcpdump utility is not appended in the packagegroup-rdk-oss-broadband.bbappend file in the Raspberry-pi build whereas you can find the tcpdump utility appended for the corresponding file for Emulator build.
- So, if we include tcpdump utility in the above mentioned file, libpcap. So file will be present in the rootfs of the raspberry-pi build.
- See below output from the raspberry-pi device after adding tcpdump -

root@RaspberryPi-Gateway:/# find . -iname libpcap* ./usr/lib/libpcap.so.1.7.4 ./usr/lib/libpcap.so.1
Currently no such pre-built images are available for download. You can follow the simple build instructions to generate your own build.

Eg:

- RDK-B Raspberrypi - Host SetUp and Build Instructions
- RDK-B Emulator Build Instructions

The eRT (embedded router) subsystem is generally used to perform dmcli operations on Raspberry pi which can function as a basic router. To perform dmcli operations in emulator we use simu.
We're simply building the application using go build (without any Yocto recipes) as shown below -

env CC=arm-linux-gnueabihf-gcc LD=arm-linux-gnueabihf-ld GOOS=linux GOARCH=arm GOARM=7 CGO_ENABLED=1 go build

- RDK officially doesn't support GO language compatibility. But you may always try GO binaries as most RDKM code bases are based on Yocto open-embedded builds and the binaries work out of the box.
- You can simply build the application using go build (without any Yocto recipes) as shown below -
  - *env CC=arm-linux-gnueabihf-gcc LD=arm-linux-gnueabihf-ld GOOS=linux GOARCH=arm GOARM=7 CGO_ENABLED=1 go build*

- Please refer to https://code.rdkcentral.com/r/plugins/gitiles/rdkb/components/opensource/ccsp/CcspWifiAgent/+/refs/heads/rdk-next/config-atom/TR181-WiFi-USGv2.XML. This gives objects, parameters list for WiFi along with functions call. The XML file (TR181-WiFi-USGv2.XML) file provides details regarding the data models available under Device.WiFi.
- . In older releases , the XML files are available under /usr/ccsp/<component> . But with the latest code, instead of reading and parsing the XML at runtime, the new approach includes conversion of XML to .C/.cpp with the help of XML2C and creating a shared library (libWi-Fi.so). These changes are done for rdkb to reduce image size and improve boot times.

Please refer https://wiki.rdkcentral.com/display/RDK/RDKB+Containerization+in+RPI+-+User+Manual+-+2019+M3 .

? Unknown Attachment

Below are the two ways to enable debug logs for component

1. adding prints in the respective place of code
2. using rdkb logger(LogAgent) .
   LoggerEnable - to enable/disable logs for particular component. If it is TRUE logs are enabled otherwise logs are disabled.
   LogLevels - To set different log levels for each component. By default all modules log level is 4 (RDK_LOG_INFO).

The steps are –

- Port valgrind to the platform (if not already available)
- Compile the component with debug symbols enabled (i.e, avoid stripping the component executable/library. alternatively, you may replace the stripped binary with unstripped binary if it is not a read-only image)
- Stop the service and start the binary as part of valgrind (just like we load any normal binary using valgrind)
- Once you have enough data collected (or observed the issue you were trying to debug), you may stop it and examine the xml file for details.

For reference please look into this attachment :

? Unknown Attachment

- ipset is available in real RDK-B devices(like the XB6 devices for example). To enable ipset for RDKB emulator, you can do that as the yocto recipes for ipset are already available at /meta-openembedded/meta-networking/recipes-extended/ipset/38.bb
- You need to add the recipe to the package group for emulator to use it up during build time (you may add them in ./meta-rdk-bsp-emulator /recipes-core/packagegroups/packagegroup-rdk-ccsp-broadband.bbappend)

Follow the below steps to customize kernel –

- Need to Create config file within linux bb/bbappend file available directory
  Path: meta-raspberrypi/recipes-kernel/linux/linux-raspberrypi
  (XXX.cfg) - Example:camera.cfg
- Configuration parameter addition: Add config data to created config file so as to enable it. For example, add the below line to our camera.cfg file
  CONFIG_VIDEO_BCM2835=y
- Need to add created config file under SRC_URI of linux bb/bbappend file mentioned in step 1
  SRC_URI = "file://camera.cfg"
- Do compilation and Installation
  bitbake "component-name" -c compile -f
  bitbake "component-name" –f

Yes, igmp_snooper program is present in RDK-B .

- There is a shell script named "*service_mcastsnooper.sh*" under */etc/utopia/service.d* directory, and in the script, it will execute "igmp_snooper $sw_opt $if_opt $querier_opt

The data model can be managed via TR-069 , TR-181, management protocols like SNMP, WebUI, WEBPA .

- The eCM and eRT subsystems will always coexists in a **DOCSIS based WAN frontend gateway devices**. Any request for eCM parameters will be routed to eCM Message Bus Adapter component which uses **SNMP or other protocols** to talk to the cable modem firmware in the backend.
- In Rpi, we have only eRT and in emulator we have simu.
- eCM is applicable only to DOCSIS based devices , which uses SNMP based OIDs.

Use NTP sync to run in the background to sync time with NTP servers.Update the NTP servers in ntp conf file if needed.
Minimum Hardware requirement :

- A Raspberry Pi 3B with the below configuration can be an ideal minimum requirement to support most of the functionalities for RDK-B.
- The configuration of RPi 3B is CPU: Quadcore 1.2 GHz Broadcom BCM2837 64bit with 1GB RAM.
- R-Pi 3B would be sufficient for the basic wireless functionality using 2.4GHz. In case of 5 GHz band support R-Pi 3B+ would be good (Note : R-Pi 3B+ doesn't support simultaneous dual band operations).
- Flash: The minimum size is around 285 MB. Anything above this should work just fine for loading/storing the build.
- RAM: As far as the logs/database is concerned, anything around 16 MB should be sufficient.

The above requirement supports all the basic functionalities of RDK-B.

- RPC download is not supported in RDK-B.
- RDK-B never had "cisco.spvtg.ccsp.fu.Configuration". This part of the code has come through initial code drop and is deprecated.

Yes. RDKB has the below options to modify SSID name, password, security protocol etc.

- A WebUI, where you can modify these items just like the UI for most normal routers.
- 'dmcli' command line utility which will execute TR69 commands to modify the mentioned items.

Yes. You can invoke the data models via dmcli commands from your service or via Cdm_GetParam set of APIs from your code. The TR-181 data model specifications that are available in the TR-181 data model XML for each component can be referred.
The CA certificate file and device certificate/private key can be configured in ccsp_tr069_pa_cfg.xml for https support.
There are already reserve SSID for Guest WiFi network. Just enable the SSID using the respective DM to get the features in place.
Duktape is a lightweight javascript engine (https://duktape.org/). Alone it does not provide a web development engine like php or node.js. Therefore,on top of duktape there is a templating engine(called jst) and web server api. To make migration as easy as possible the style of templating and api signiture will match php as closely as possible. Many php functions and variables will be rewritten in javascript, so that changes to the exist code is minimized.
It actually defines what kind of Internet traffic is allowed or blocked. For more details, Please Refer : Firewall - Rule persistence
Please refer Integration Guide for third-party applications into RDK-B stack .
Use journalctl -xu parodus or can look for parodus.log in the log folder i.e. /rdklogs/logs/.
In <image>.bbappend file , upgrade python version using ROOTFS_POSTPROCESS_COMMAND
Code :

```
#python upgrade
IMAGE_INSTALL_append = " python3 "
ROOTFS_POSTPROCESS_COMMAND += "enable_python3; "
enable_python3() {
ln -sf /usr/bin/python3 ${IMAGE_ROOTFS}${bindir}/python
}
```

On compiling the code and flashing the image , python3 will be available in your board . Check with below command.

```
root@RaspberryPi-Gateway:~# python --version
Python 3.5.2
```

Checksum mismatch error is related to Fetcher failure for URL . The ways to fix this error are as follows

1. the url might be deprecated . Hence should use the right URL with right path
2. md5 checksum must be pointing to older one , which should be replaced with the latest checksum
3. if the component is not required (or not maintained) , then they can be masked from inc file

- For this you need to convert all C/C++ logging statements(like printf) to use standard logging functions CcspTraceInfo, CcspTraceWarning, CcspTraceError etc. CcspTraceXxxx functions are based on open-source 'log4c'.
- By using the above apis the logs are automatically written to logfiles.
- After conversion to standard logging functions ,you can verify the logs under /rdklogs/logs/.