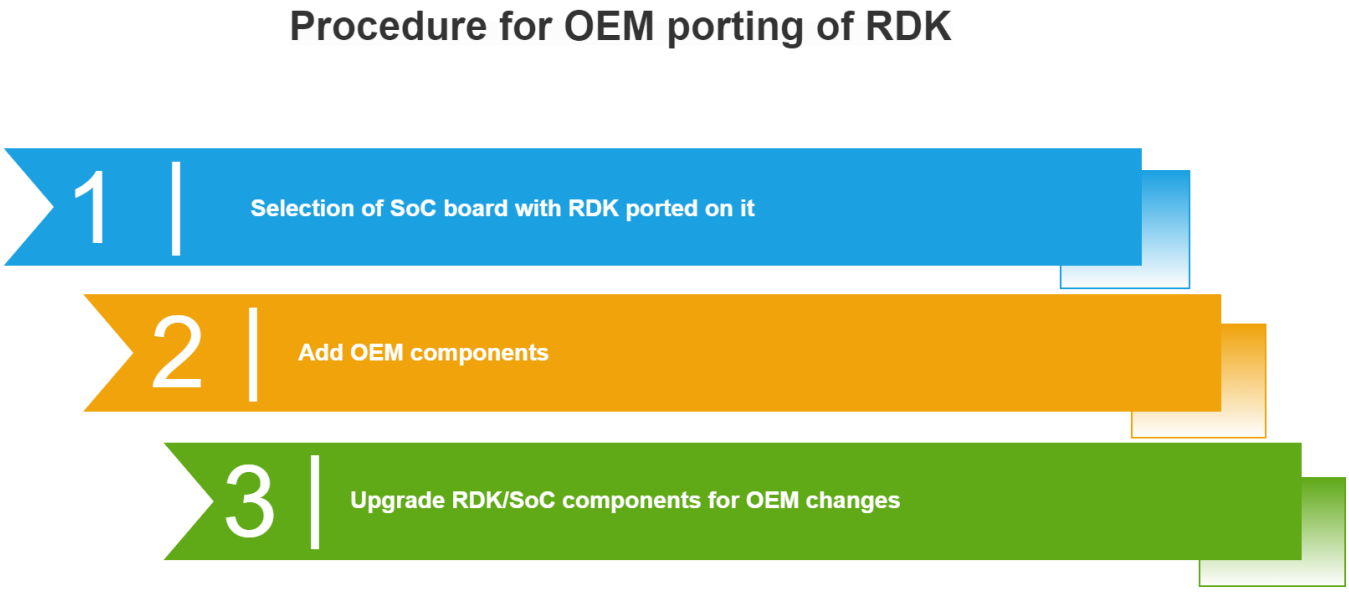
RDK-V 4.0 - OEM Porting Guide

The aim of this document is to explain the procedure for OEM porting of RDK.



Step 1 : Selection of SoC board with RDK ported on it

Refer this page SoC Platform Firmware to know the details about Yocto manifests, SoC meta-layer creation includes adding the Machine Configuration File for the new SoC .

Step 2 : Add OEM components

OEM needs to add OEM specific components like Firmware Upgrade, Secure Boot Loader, MFR libraries, Vendor Specific Information, NVRAM files and partition, Provisioning, OEM Specific drivers, STB Utilities, RDK Device-Specific Patches, Image Generation Utiities etc. as well as interfacing layers to the generic RDK for relevant OEM code modules (see below)

Step 3 : Upgrade RDK/SoC components for OEM changes

Any Revision change in SoC layer is usually done by SoC's build environment and the new SDK or revision is updated in recipe. If a new recipe is added for any update in SoC software, then can be handled using PREFERRED_VERSION Yocto flag in meta layer

Components of OEM Interface

Bluetooth

Bluetooth Manager implements the Bluetooth HAL i.e. Bluetooth Core (BTRCore) API. Bluetooth HAL interface provides a software abstraction layer that interfaces with the actual Bluetooth implementation and/or drivers. RDK Bluetooth HAL layer enables projects to pick any Bluetooth profiles as per their requirements. Bluetooth HAL uses BlueZ5.42 stack which is a quite popular Linux Bluetooth library.

- Bluetooth HAL - Provides APIs to perform Bluetooth operations by abstracting and simplifying the complexities of Bt-Ifce (& BLuez)
- Bt-Ifce - Abstracts Bluez versions and serves as a HAL for other Bluetooth stacks - Interacts with Bluez over DBus
- Bluez - Interacts with kernel layer bluetooth modules

Modules

- Bluetooth Core APIs
- Bluetooth Core Types

Crash Upload

- Uploads core dumps to a FTP server if there are any
- This interface is optional, OEM may implement a customized script for uploading the crash dump files to a server using specific certificate files

Device Settings

- OEM implementation includes device specific configuration parameters & customizing data structures for front panel setting and so on

DTCP

- Integrates the SoC provided DTCP library with DTCP/IP manager Interface implementation which Manages source/sink DTCP/IP sessions and performs the encryption/decryption
- Provides setup scripts to initialize the default DTCP data to device specific persistent partition

hwselftest

Provides platform specific configuration options for Hardware test. Which will run periodically in background to check attached hardware health.

- OEM dependency is limited to providing a configuration file hwselftest.conf which defines the hardware interfaces that should be tested e.g. partition name, HDMI, IR, MOCA and so on

LED Manager

LED Manager is used to control the LED patterns during different system events.

- OEM should implement the template code to ensure that it matches with their LED coloring scheme and other configuration parameters such as brightness level and number of LED types, multi-color or single color LEDs

tenableHDCP

This handles the HDCP service operations such as enable or disable the HDCP.

- OEM interface includes implementing manufacturer specific calls to read the keys and so on

Wi-Fi

- OEM specific Wi-Fi driver configuration files and utility library can be added optionally

For details on the Wi-Fi HAL Public APIs and Data Types, please refer: https://wiki.rdkcentral.com/doxygen/rdkv-opensourced/df/dce/group___w_i_f_i___h_a_l.html