For Operators

Before You Begin

RDK License

Operators are advised to get into an agreement with RDK Management LLC to obtain the free license so as to use the complete RDK Code base in their platform. More details about license is available at https://rdkcentral.com/licenses/ . Please email info@rdkcentral.com if you have additional questions about licenses or membership

On this Page:

Overview

This page provides the details and guidance to the Operators on how to adopt RDK. The step by step procedure for an operator to get an RDK based Platform up and running is also described in detail.

Error rendering macro 'excerpt-include'

User 'null' does not have permission to view the page.

Product Specifications

The first step to get a fully functional product is to define the product features and see if they meet the standard requirements. See here to know what are all the features available in RDK-V and can implement based on your requirement. Operator can use this as a guide while engineering the RDK Operator platform.

Device Firmware

Operators can make use of the details available below to start developing a Yocto build to do the final additions of Operator specific changes to the device. This will help Operator to add their own final product features as well as Operator specific patches/changes.

Yocto Manifests

Yocto based RDK builds are flexible enough to easily accommodate SoC / OEM / Operator / third party changes. The starting point for the Yocto builds are a manifest file. The manifest file is an xml file that contains details of the different open embedded Yocto build layers, meta layers , rdk as well as open source components that needs to be fetched during initial stages (than during bitbake time) as well as the URL locations from where the data can be pulled. A set of sample manifests that can be used as a template for developing Operator specific manifests are given below

#	File	Remarks
1	Device Specific Manifest	This is the starting point of the build and is specific to a device/board . If the SoC/OEM has only one target device /board, still it is recommended to maintain a different manifest for SoC/OEM for keeping it future-proof. Usually it contains references to other manifest files which will be having specific set of repos
2	SoC Manifest	This manifest contains meta layer details of SoC and , optionally, some SoC specific repos
3	OEM Manifest	This manifest contains meta layer details of OEM and , optionally, some OEM specific repos
4	Operator Manifest	This manifest contains meta layer details for operators and , optionally, some operator specific layers.
5	OE layers Manifest	This manifest has details of the basic Yocto Open Embedded layers
6	RDK-V Manifest	This manifest can contain meta layers specific to RDK & RDK-V and , for EXTERNALSRC cases, the RDK & RDK-V component repo details too. It might be supplemented with a conf file too
7	Third party Apps manifest	This manifest can contain meta layers related to premium streaming applications which are chosen by Operator

The default manifest repo in RDKM Git is at https://code.rdkcentral.com/r/#/c/manifests/ . Operator need to have their own manifest file in this location which will be mentioned in the 'Device specific manifest' - which is the starting point of builds for a particular target device

Operator meta-layer creation

To match the layered structure of Yocto builds, a Operator specific layer is used to include Operator changes and additions. Use the yocto-layer create sub-command to create a new general layer.

```
$ yocto-layer create mylayer
```

There shall be separate device (machine) configuration file (.conf) for each device for the particular chip family for which the layer is intended for. The general naming terminology for Operator layer is meta-rdk-<soc name>

The device (machine) configuration file shall include corresponding include (.inc) file to get machine configuration details.

Adding the Machine Configuration File for the new Operator

Each meta-* layer should have a conf folder containing the machine configuration we can select during 'source setup-environment' . Optionally it can also have a class/classes folder for keeping information that is useful to share between metadata files.

To add a machine configuration, you need to add a .conf file with details of the device being added to the conf/machine/ file. In the machine conf file the basic machine configuration should be defined.

Once the conf files are in place, the layer details need to be added to the corresponding package group file. Based on platform, the package group file will be grouped into multiple files with one bitbake fiile and multiple append files. Operator can add the Operator layer details to required package group append files applicable to the target device build to get the features into the final build

Operator Specific Apps

Operators will be having a range of Operator specific applications from simple generic device information apps to Operator specific content applications. RDK's Yocto based layered structure allows Operators to easily integrate, upgrade and maintain their apps in their RDK based IP Set-top devices. For more details on App support in RDK, please refer Applications(only for RDK licensee) as well as Operator Platform Firmware for the engineering details.

Operator Specific UI

RDK supports usage of multiple User Interface in the RDK IP Set-top devices. Operators can choose from among the already available UIs that are available with RDK as well as develop and use their own UI.

Follow the below steps to get Lightning UI running in devices:

Clone and Build Instructions

This section details about how to customize the current UI source code and develop further.

Follow the commands to run the application on you laptop.

- Firstly install Lightning CLI
- Type lng to confirm the installation. You will see the below output.

```
$ npm install -g @lightningjs/cli
$ lng
Usage: index lightning-cli <command> [options]

Options:
  -V, --version   output the version number
  -h, --help      display help for command

Commands:
  create - Create a new Lightning App
  build - Build a local development version of the Lightning App
  serve - Start a local webserver and run a built Lightning App in a web browser
  watch - Watch for file changes and automatically rebuild the App
  dev - Build a local Lightning App, start a local webserver, run a built Lightning App in a web
browser and watch for changes
  docs - Open the Lightning-SDK documentation
  dist [options] - Create a standalone distributable version of the Lightning App
  upload - Upload the Lightning App to the Metrological Back Office to be published in an App Store
  update - Update the Lightning-CLI to the latest version
  help [command] - display help for command
```

- Clone repo

```
$ git clone "https://code.rdkcentral.com/r/components/opensource/RDK_apps"
$ cd RDK_apps
$ git checkout rdk-next
$ cd accelerator-home-ui
Note: if you are not able to launch offline UI, add .env file inside accelerator-home-ui/ and
add 'LNG_BUNDLER=esbuild' to force the bundler. Make sure you have esbuild dependency (npm install
esbuild)
$ npm install
```

- Build and Host

```
$ lng build
$ lng serve
```

- App will be hosted on http://127.0.0.1:8080

- The RDK Accelerator Home UI version 3 is the latest version (3.7) of the RDK reference UI which provides a full featured UI with a modern minimalistic design using a dark theme.

  - [x] New Feature: UIv3.7 Added multi-profile support.

- This single UI currently supports IP-STB, Hybrid-STB and TV profile RDK stacks. UI uses DTV Plugin and HDMIInput plugins to distinguish the platform profile; UI will consider the platform as TV if stack has HDMIInput plugin support, Hybrid STB if DTV plugin is present else the default IP-STB.
- The UI uses Lightning SDK Plugins such as Language, Router, Storage and supports Premium apps such as Amazon, YouTube, Xumo along with a selection of Lightning apps from the Metrological App Store. It also includes a wide section of settings which allow a user to take full advantage of the device features such as Audio, Video, Network, Language and more.
- To use and to know more about RDK Accelerator Home UI please refer Accelerator Home UI - v3 .
- Based on resident app reference implementation Operator need to bring up the UI.

What operators need to do if they plan to use Accelerator UI as such (or with minimal changes)?

1. (Optional): Create an offline and online version of the UI.
    a. Offline can have minimal things (like pair the remote, settings to connect to WiFi etc) which is available even if device is not connected to Internet.
    b. Online will be the full fledged UI and will be fetched from a server with latest version
2. Host their UI in their own production server and make RDK use it. For this , operator need to change the URL in this line https://code.rdkcentral.com/r/plugins/gitiles/rdk/components/generic/appmanager/+/refs/heads/rdk-next/residentapp/residentApp.sh#165

What operators need to do if they plan to use a different UI?

1. Operator can develop their own UI either using Html or Lightning. Below section stall describe how a Lightning UI can be deployed.
2. ResidentApp service serves the offline pages from DOCROOT of webserver(lighttpd) running on device. Operator need to install the UI assets in the DOCROOT and that will be launched by the https://code.rdkcentral.com/r/plugins/gitiles/rdk/components/generic/appmanager/+/refs/heads/rdk-next/residentapp/residentApp.sh#166 when the device starts up.
3. To make the device start always with offline UI pages; remove or comment out https://code.rdkcentral.com/r/plugins/gitiles/rdk/components/generic/appmanager/+/refs/heads/rdk-next/residentapp/residentApp.sh#230.

App Support

Along with the Operator specific apps, Operator can support a lot of generic apps in RDK IP Set-top devices by taking advantage of RDK Support for Native as well as Web Apps in IP based Set-top platforms. Operator can easily port native apps in their platforms (for some third party apps, Operator need to obtain certification from those third party) or can host their own app store and then use Web Apps to show content. For more information on App support in RDK, please refer Applications (only for RDK licensee).

Factory Test App

The Factory Test App is a special standalone application used for production line testing at the TV assembly factory of the OEM manufacturer. It is embedded directly in the firmware image that gets flashed onto the boards on the TV production line.

<ToDo>

Provisioning Support

Operator needs to add provisioning support in device so that Device provisioning can be done once deployed at customer premise. The steps for this varies based on platform as well as Operator type.

Provisioning support refers to the scenario such as when you launch Sample app on your mobile it takes to login page and so you have to login there, account there i.e. actually the app has to authenticate your login.

Disaster Recovery

Disaster recovery is an inevitable part of the CPE life cycle. Operator, based on their disaster recovery strategy, could add support for this in the device. While there are some generic guidelines followed across industry, there is no single step that works for all. Operators could easily add their business logic to RDK as part of Operator firmware engineering as described in Operator Platform Firmware.

As an operator they have to handle crashes/disaster happens and support any factory reset like OTA upgrade.

Test & Certification of devices

Once the device engineering is completed from Operator side, the device can be test and verified easily by the Certification support provided by RDKM. Details of coverage as well as major cases are explained here (only for RDK licensee) .

Certification suite is available at RDK IP Set-top Product Certification (only for RDK licensee) and for TDK test app please refer TDK-V Documentation.

In short, to get an RDK based device to field, Operators need to get

- RDK license
  - Operator has to get a RDK license and then any premium apps porting operators should have a project agreed with the corresponding premium app like for youtube it is google to access premium app/any app's plugin or code implementation for app.
- In case Operator is planning to include third party apps (like Netflix , Amazon Prime etc. to name a few) there should be a Commercial License Agreement between operators and premium apps.
- Project agreement with OEM's/ SoC

What Operators get from RDK:

- Pre-certified stack for third party apps is available in RDK as a ***Common Port Architecture.*** For details see Common Port for Native Apps.
- RDKServices are available to customize RDKService plugin plugout components for all the stacks mainly Bluetooth, Wifi and front panel, power manager, device settings etc, pre-set apps and basic architecture of OTA are available.

What Operators need to do:

- As an Operator, the certification for the operators device has to be done by them(eg: for premium apps).
- Customization of boot loader for operator specific , operator specific UI, Operator specific any customization on the stack such as Power manager(Power specifications to Set-top box to consume only specified power)
- Some Customization of remote and operator specific operations such as other configurations settings and their persistent areas(like premium app logo size on operator device).
- UI integration with RDKServices and UI Specific customizations
  - Operator specific task on RDK are customization of UI to adopt RDKServices so as part of customization of UI they can refer the resident app implementation. As part of customisation of UI,OTA upgrade is one of the first main point .As an operator they need to get the OTA backend server . Xconf is the RDK recommended OTA method.It is not mandatory this can be optional(Like the firmware update plugin is available there readily they can directly use that infrastructure otherwise they need to configure their headend with the xconf server.)
  - Operator specific requirements has to be implemented by operator (eg.frontpanel related or any customisation of UI and any such requirements)