REFERENCE DESIGN KIT FOR BROADBAND DEVICES

Discover the Future of Residential Gateway Software

technicolor

FEEL THE WONDER

REFERENCE DESIGN KIT FOR BROADBAND DEVICES

Discover the Future of Residential Gateway Software

Reference Design Kit for Broadband devices (RDK-B) is achieving for residential and small business gateways what Reference Design Kit for Video devices (RDK-V) has accomplished for set-top boxes. Its goals are to bring a portable, modular, and component-based gateway software stack to the market and redefine the future of residential gateway software.

RDK-B is:

- Eliminating the need to manage a monolithic stack
- Making it easier to add new features and quickly create product variants
- Supporting multiple hardware platforms.

Residential gateways offer a path to greater subscriber value and differentiation. Using RDK-B, operators can take full advantage of these gateways to introduce new services and derive new revenues faster.

Contents

Introduction

Reference Design Kit for Broadband devices (RDK-B) is achieving for residential and small business gateways what Reference Design Kit for Video devices (RDK-V) has accomplished for set-top boxes. RDK-B deployments are ramping up quickly on new gateway devices, as well as supporting gateways already deployed and capable of running the RDK-B stack. Although RDK-B is a relatively new gateway stack for the industry, it is based on a proven stack deployed on millions of residential and small business gateways.

The baseline for RDK-B was contributed to open source by Cisco Systems, Inc. (now Technicolor). It has been ported to other original equipment manufacturer (OEM) platforms—and is being integrated on multiple system on a chip (SoC) solutions.

Although RDK-V and RDK-B both include the term “Reference Design Kit” in their names, they are different software stacks. The RDK-V architecture and functionality are focused on complex video functions such as tuning, conditional access, digital rights management (DRM), and stream management—functionality that is central to video devices such as set-top boxes.

The RDK-B architecture and functionality are focused on complex broadband functions such as wide area networking, local area networking, management and diagnostics, and home-networking interfaces, such as Wi-Fi and Multimedia over Coax Alliance (MoCA). This functionality is central to broadband devices such as residential and business gateways.

Although the stacks are different, the purpose of standardizing these separate stacks is the same. Both RDK-V and RDK-B help accelerate the deployment of next-generation services by multichannel video providers (MVPDs) and broadband Internet service providers. Standardizing the stacks allows service providers to standardize certain elements of these products and devices, such as set-top boxes, gateways, and converged devices, and more easily customize the applications and user experiences that ride on top. These stacks sit below the service provider application and services layer and provide a common interface to the underlying SoCs, allowing the same stack to be used across multiple hardware platforms.

Why RDK Is Needed for Broadband Devices

With the convergence of all services over IP, including video and broadband services, and the evolution to the Internet of Things (IoT), the residential gateway is becoming more central to service delivery, both to the home and within it. By leveraging the gateway's advanced hardware and software capabilities, service providers can introduce new services and rapidly generate incremental revenue streams. The residential gateway allows new devices to be added to the home network, while giving the operator greater insight and access into those networks.

As the demarcation point for all IP-based services delivered to the home or small business, the gateway plays a central role. It is crucial for service delivery, as well as an important tool for customer support representatives to help diagnose and resolve issues when customers call. However, some of the gateway value has not been realized, because the necessary software varies from OEM to OEM—and at times, even between different devices from the same OEM.

As a result, deploying new services required multiple parallel efforts across different hardware and software platforms. In addition, system issues are unique to a particular platform. That means fixing "bugs" on one platform won't help resolve issues on separate hardware and software platforms, because their problems will be different. Management interfaces across devices also vary, which makes it more difficult for technicians, customer support representatives, and even subscribers to manage these devices and solve problems when they arise.

Standardizing the gateway software stack drives operational efficiencies as well as engineering efficiencies, so service providers can deploy new innovative services faster, simplify installation and support for these devices and the home network, and make the most of a single software effort across multiple hardware platforms. In addition, by using open sourcing for this stack, the full industry is empowered to drive innovation into the stack, which leads to a more full-featured and robust stack that can be used across a range of hardware platforms, with no hard dependencies on OEM or SoC. The RDK-B vision is enabling the full potential that the market sees for residential and small business gateways.

Several options were considered before the current stack was selected for RDK-B. The criteria to consider for a residential gateway stack for the service provider market include:

- Available through open source
- Multivendor support
- Remote management using industry-standard protocols such as TR-069, TR-181, and Simple Network Management Protocol (SNMP)
- Integrated flexible control plane
- Software component-level modularity
- Support for downloadable applications
- Wi-Fi hotspot services support
- Advanced service provider features, such as traffic isolation using GRE tunnels

The baseline for RDK-B is based on a proven Technicolor (aka Cisco) stack that is deployed on millions of service provider residential and business gateways. A majority of the criteria listed here are already supported in the code that Technicolor has contributed to open source. RDK Management LLC evaluated the options for RDK-B and selected a stack that met the architectural vision for RDK-B and was already field proven in a service provider environment.

Now that this stack is freely available as open source, multiple contributors are working to enhance the software. This process accelerates feature velocity and helps to ensure that this stack continues to grow and evolve to solve the problems faced by service providers today and in the future.

The RDK-B Open Source Model

The value of an open source software model is well documented, so it is important to note that RDK-B has been launched via open source. Historically, RDK-V has followed more of a shared-source model that is only available to licensees. RDK-V generic source code has only been available for organizations that have signed a license. RDK-B uses a true open source model where all of the generic RDK-B components are available through open source. RDK Management LLC hosts the software and it is freely available at code.rdkcentral.com.

In addition to the open source RDK-B software, there are layers provided by SoC vendors and layers provided by OEMs, which are required for device-specific functionality. When combined with the generic RDK-B components, these complementary software elements (provided by SoC and OEM vendors) create a fully functional gateway device. This more truly open source model for RDK-B allows a wide community of developers to identify bugs, make suggestions, and submit new code, so the RDK community can enhance and evolve the stack more rapidly.

RDK-B components that have been released to open source are distributed under the Apache Software License (ASL) Version 2. Various open source licenses were considered before ASL was finally selected. Widely used in the open source software community, ASL has been approved by the Open Source Initiative. It is a permissive license, conducive to commercial development and proprietary redistribution, and it is worth noting that Android is also distributed under ASL.

Many companies prefer permissive licenses like ASL, because they make it possible to use open source software code that encourages, but does not require, authors to turn proprietary enhancements back over to the open source software community. This license encourages commercial adoption of open source software because it’s possible for companies to benefit from investing in enhancements made to existing open source software solutions. Making RDK-B components available under the ASL should allow a broader number of companies to adopt the platform and build on it without having to expose the inner workings of proprietary technologies. The selection of ASL for RDK-B distribution helps address the RDK community’s need to have a truly open source solution, while also encouraging commercial development on top of the platform.

RDK-B Device Types and Usage

While RDK-B was deployed first on DOCSIS® residential voice gateways, it is in no way limited to DOCSIS gateways. Any device that includes IP data networking functionality is a potential target for RDK-B or its components. The modular, component-based architecture of RDK-B makes it simple to target multiple platforms with the same stack. Some products types that can benefit from RDK-B include:

Home gateways

While offering the functionality of basic modems and routers, these products also offer more robust capabilities to provide reliable voice and high-speed Internet services to subscribers. Service providers that want to offer value-added services often deploy these gateway devices. High-end gateways also include software and hardware that support additional services and capabilities, such as voice over IP (VoIP) and Internet Protocol television (IPTV) services. These devices can also include home networking interfaces such as Ethernet, Wi-Fi, MoCA, HPNA, and HomePlug.

Small business gateways

These products are similar to home gateways and, in fact, may often use the same hardware to deliver data services to small businesses. Specific gateway features are provided through software on the device. For example, a Layer-2 Virtual Private Network (L2VPN) feature allows service providers to offer a Layer 2 transparent LAN service to commercial enterprises.

Wi-Fi extenders

These devices provide additional Wi-Fi access points to improve coverage in a home or business. MoCA or other networks may be used to interconnect these devices to at gateway and bridge traffic to the Wi-Fi network. As Wi-Fi becomes more prevalent in all connected devices, improving Wi-Fi coverage in homes and small businesses becomes more important. MoCA-to-Wi-Fi extenders use the in-home coaxial cable for highly reliable connectivity to bridge high-speed data from the MoCA network to the Wi-Fi interfaces on these devices. Other Wi-Fi extenders can support Ethernet or Power Line networking. This effectively extends the Wi-Fi functionality from the gateway to the extenders and improves Wi-Fi coverage throughout a home or business

IoT controller

These products support services such as home security, home monitoring, and home automation over a network. For example, users can access automated lighting, heating, and security systems over the Internet while at work, running errands, or on vacation. The products act as a bridge between the devices that are being controlled in the home to the Internet. IoT controllers also allow service providers to offer new packages and generate new revenue streams. This controller function can reside in a stand-alone device that sits behind a residential or small business gateway, or it can be integrated into the gateway for greenfield installations.

This list of example products is not exhaustive, and it would not be surprising to see RDK-B running on other devices that include data networking functionality. In addition, even though the initial focus of RDK-B has been targeted at cable gateways, there are no dependencies that will limit RDK-B to cable products. So it is likely to be used in other markets as well.

RDK-B Goals

Technicolor's contribution to RDK-B began with an internal effort to develop software used on Technicolor residential and business gateways. The original goals for this software are now similar to the goals of RDK-B. They include:

- Software modularity
- Abstraction of external management protocols
- Independence from wide area network type
- Silicon independence
- Linux kernel independence
- Software structure that allows multiple organizations and teams to work in parallel

Architecture Overview

This high-level architecture overview describes how the software achieves its goals.

Software Modularity

The RDK-B software is composed of many different software components. Originally, the software component name was chosen to represent a software technology that had all the same advantages of hardware components used to create hardware designs. For example, these hardware components included memory integrated circuits (IC), CPUs, interface electronics, and similar off-the-shelf building blocks that can be used to create many different types of hardware products. To make these hardware components reusable, several concepts are needed. In a slightly more abstract form, many of the same concepts are needed to make software components reusable.

The following component reuse concepts are included in RDK-B:

- RDK-B software components have well-defined interfaces. This is analogous to electronic components having well-defined I/O balls or pins.
- RDK-B software components are reusable in multiple designs without changing the component itself. This is analogous to a CPU, memory IC, or electronic interface component that can be used in multiple hardware designs.
- RDK-B software components can be designed and created by one team and used by other teams (just like an electronic component).
- RDK-B software components can be released independently, with their own software versions. A top-level software source code release does not have to contain all the source code as a part of the release. Each top-level release may simply refer to the software components that are needed and the version number for each.

These software components are structured to use a common inter-process communication method for control-plane messages such as setting a parameter, getting status, and eventing. This method is organized around a common software bus. Linux DBus was chosen as the underlying framework to allow components to plug in and communicate with each other. While it provides a good framework to build on, DBus needed some additional features for true component modularity, including:

- Component-level service discovery: This feature is provided by a software component known as Component Registrar, which allows other components to register the internal data model objects that they support and the DBus path used to reach these objects. Then, when other components need to determine what features are available in a particular software build — and discover how to access these features — Component Registrar provides this information. Through these capabilities, this mechanism provides a flexible way to build a software system that can use any number of software components, allows components to discover what features are available and how to access these features.
- A standard internal data model object structure: TR-181 was used as the standard data model object structure for internal communications over the common control-plane software bus. TR-181 has a number of advantages, including well-defined open and published specifications, as well as an extensible infrastructure used to extend the existing data model when necessary.

  Note: *The external data model does not have to use TR-181. This section only describes the internal data model used over the common control-plane software bus. The concept of external data model abstraction is discussed later in this white paper.*

- A common abstraction layer and common set of primitives to use DBus: The Common Lib component provides a layer on top of DBus. This component provide a standard set of mechanisms for all other components to use. This layer also provides an abstraction layer that would allow the underlying IPC framework to be swapped out in the future, if needed, without requiring changes to other components.

- Multi-CPU and multi-kernel abstraction: Linux DBus provides a framework that components can use to communicate within the same Linux kernel domain. A means was needed to support low-cost, embedded devices that use multiple CPUs of different types, which may run different kernels on each CPU. The DBus abstraction layer and component-discovery functions were designed with this in mind. The Common Lib includes DBus extender software that allows DBus to be transparently extended to multiple independent kernels running on independent CPUs (even CPUs of different types). The Component Registrar design also includes a mechanism to specify a network path to each CPU to allow multi-CPU and multi-kernel service discovery and messaging.

  Note: *While control-plane messaging uses the Common Lib and the common software bus to exchange messages between components, this is not intended for high-bandwidth broadband connections. High-bandwidth connections typically use direct kernel interfaces (Example:, logical Ethernet software interfaces).*

Abstraction of External Management Protocols

One original software goal was to support a flexible approach that would allow product software teams to choose which external management protocols to include in each software build. This approach had to be supported in a way that is transparent to all the internal core software components. To accomplish this goal, each choice of external management protocols is supported using a type of software component called a Protocol Agent.

Protocol Agent examples include:

- TR-069 Protocol Agent
- SNMP Protocol Agent
- Web-based user interface Protocol Agent
- HNAP Protocol Agent
- Other future Protocol Agents

The role of the Protocol Agent is simply to abstract the external management interface protocol (Example: SNMP) from all the other components.

The Protocol Agent may also provide a data model “mapper,” which is used to abstract the external data model from the common internal data model used over DBus. For example, the SNMP Protocol Agent implementation includes a flexible XML-based mapper file. It defines the mapping between SNMP management information bases (MIBs), which are used to manage the device, and the internal data model objects used by the core components. This mapper file concept allows the software product teams to easily edit the XML file to match the custom external MIB definitions that are needed by different service providers.

Of course, not every software build (or every product) will need every Protocol Agent. The software product team only chooses the Protocol Agents that are needed for a specific product. For example, if a product doesn’t need to support SNMP, then the SNMP Protocol Agent is not included in the software build for that product. (This concept mimics the flexibility offered to a hardware product designer. Many types of CPUs and memory components are available, and the designer can select one or more of these hardware components to include in a particular design.)

Independence from Wide Area Network Type

Gateways using RDK-B today may include a DOCSIS wide area network (WAN) interface, but RDK-B is designed to work with any WAN type. In fact, RDK-B does **not** include a cable modem DOCSIS network stack. It only includes a component that provides a Cable Modem (CM) Agent. The function of the CM Agent component is to provide an abstraction layer known as a CM HAL (Hardware Abstraction Layer). The CM HAL provides a well- defined software interface that cable modem software developers can use to interface to RDK-B.

Other WAN agent components may be developed in the future, including Ethernet passive optical network (EPON) agents, GPON agents, and more, as long as these software components provide a clean abstraction layer to separate the WAN-specific functions from the remainder of RDK-B.

Silicon Independence

RDK-B is architected to be independent of specific silicon choices that a product team makes when creating a product. Hardware abstraction layers are defined in many components to achieve this goal.

- Wi-Fi example: If a gateway product needs to include Wi-Fi Access Point features, the product software team can include the Wi-Fi Manager Software Component at build time. This component includes a Wi-Fi HAL that can be supported by Wi-Fi silicon providers to allow their Wi-Fi drivers to easily plug into RDK-B.
- MoCA example: If a gateway product needs to include a MoCA interface, the MoCA Manager Software Component can be included at build time. This component includes a MoCA HAL that can be supported by MoCA silicon providers, allowing their MoCA drivers to easily plug into RDK-B. The MoCA HAL is currently supported by another software component, but plans call for splitting it out as a separate MoCA software component.

Linux Kernel Independence

Unlike some other open source projects, the RDK-B effort does **not** include a Linux kernel distribution. This decision was made so that SoC vendors could provide an optimized Linux kernel distribution that works best with their specific SoC implementation. Future releases of RDK-B should specify the Linux kernel revision used for testing and may also specify a minimum kernel revision.

Software Structure multiple organizations and teams to work in parallel

RDK-B supports teams that are designing and enhancing software components, as well as other teams that are using the current released version of these components to create products. This approach allows multiple organizations and teams to work in parallel. Product teams can easily use existing released components, while component teams work to add new features and functionality that will be released in a future version of each component.

Software components are also designed to use the Common Lib (and by abstraction the common software bus) for all component-to-component control-plane messaging. This design choice allows multiple teams of component developers to work together with a minimum amount of coordination and planning.

As occurs with all active software projects, the architecture and code for RDK-B is evolving. Efforts are underway to further increase software modularity, support new device types, increase performance, and add features.

Next Steps

The latest software is available at code.rdkcentral.com. This enables any vendor or service provider to use RDK-B.

As the RDK-B initiative continues to grow, next steps include:

- Expand the number of silicon providers that use RDK-B on their reference designs, as well as OEMs that use RDK-B in their products.
- Expand the number of supported WAN interfaces to include PON and other network types.
- Continue to grow the number of supported software components to enable modular solutions for multiple device types and enable software development teams to choose features to be included in each software build.
- Continue to enhance the feature set to support new service provider business and technical requirements.

As service providers work to enhance and add to the services being offered to their subscribers, many perceive residential gateways as a path to greater subscriber value and differentiation. With the release of RDK-B, operators are able to take full advantage of the gateways to introduce new services and derive new revenues more quickly than previous solutions allowed.

About the Authors

Charles Moreman

Chief Architect
Connected Home Business Unit
Technicolor

As the Chief Architect, Charles leads the overall software and hardware architecture for Technicolor's Connected Home Business Unit. In this role, he is responsible for creating new innovative architectures to solve Service Provider problems. Charles continues to be closely involved in RDK-B software architecture and helps to guide Technicolor's continuing contributions to the RDK-B project. Previously at Cisco, Charles worked as the lead architect for the software that became RDK-B and was instrumental in creating this software. In previous roles, Charles worked as the engineering director for Cisco/Scientific Atlanta Cable Modems, EMTAs and residential gateway products He also managed other software and hardware engineering teams within Cisco. Prior to joining Cisco, Charles worked as a VP of Engineering developing software, hardware and silicon technology used in the telecommunications and video compression industries.

Charles earned a Bachelor of Science in Electrical Engineering from the Georgia Institute of Technology. He has been awarded 12 US Patents.

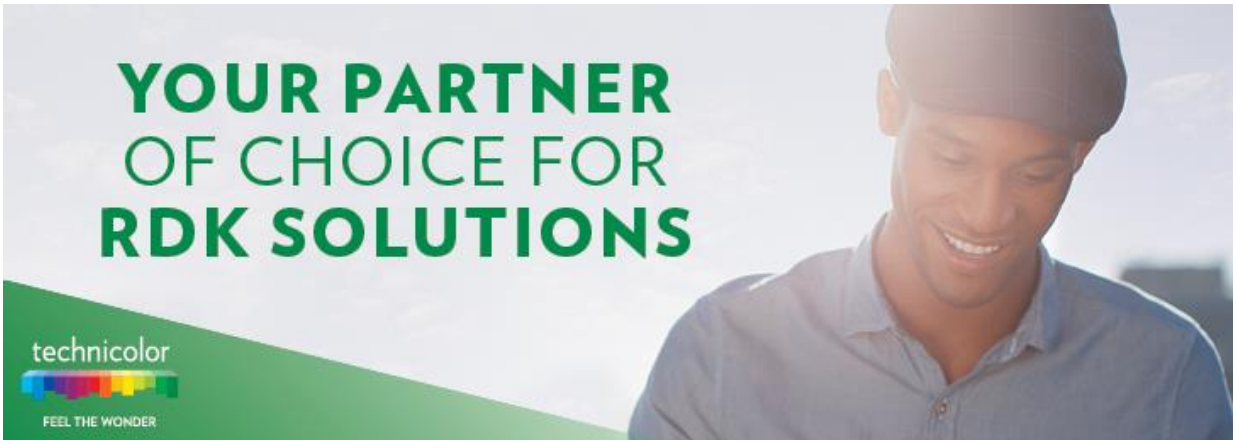
Jon Cave

VP Broadband CPE Product Strategy and Management
Technology and Product Development Group
Comcast Cable

Jon currently serves as the Vice President of Broadband CPE Product Strategy and Management within the Technology and Product Development Group of Comcast Cable. In this role, Jon is leading the team that develops innovative Broadband CPE devices used to deliver Comcast's voice, data, and cross-platform applications in customers' homes.

Jon has more than 20 years of experience in the telecommunications industry encompassing engineering, product development, and product management of data, voice, and video networking products. Prior to joining Comcast, Jon was the Director of Product Management at Cisco Systems in the Connected Devices Business Unit, responsible for defining and driving product strategy for Cisco's next generation Cable Modem, EMTA, Residential Gateway, and Unified Gateway products used to deliver voice, data and video services. Jon also has entrepreneurial experience at companies developing solutions for WiFi, power line communications, as well as VoIP products. Earlier in his career Jon also served in the United States Air Force.

Jon earned a Bachelor of Science in Electrical Engineering and Master of Science in Electrical Engineering from the Georgia Institute of Technology, and an MBA from Georgia State University.



Our RDK Philosophy

As a leader in software solutions for gateways, Technicolor (previously Cisco) created the software architecture that became RDK-B. We open sourced much of this software and it became the foundation for RDK-B. Technicolor continues to be the leader in RDK-B software through innovative new software additions and open source contributions. Our solutions and services span a number of engagement models from integration and partnerships as well as OEM supplier. We have global R&D centers of competence dedicated to software and solution development with RDK. The solutions we provide are open and modular and can accommodate your deployment needs.

About Technicolor

Technicolor, a worldwide technology leader in the media and entertainment sector, is at the forefront of digital innovation. Our world class research and innovation laboratories enable us to lead the market in delivering advanced video services to content creators and distributors. We also benefit from an extensive intellectual property portfolio focused on imaging and sound technologies. Our commitment: supporting the delivery of exciting new experiences for consumers in theaters, homes and on-the-go.

www.technicolor.com
Follow us: @Technicolor
linkedin.com/company/technicolor

Technicolor Worldwide Headquarters

1, Rue Jeanne d'Arc
92443 Issy-les-Moulineaux, France
T +33 (0)1 41 86 50 00
F +33 (0)1 41 86 56 15

technicolor.com

WP-025-V01-1611