HDMI CEC

Feature Summary

The HDMI_CEC API defines the ability to get connected HDMI devices, send messages to those devices, and to be notified when messages are received from HDMI devices.

HDMI-CEC is a protocol that provides high-level control functions between audio-visual devices connected over HDMI. CEC is a one-wire bidirectional serial bus based on industry-standard AV.Link protocol to perform control functions. All audio-visual sources are connected directly or indirectly to a display device as the ‘root’ in a tree-like structure.

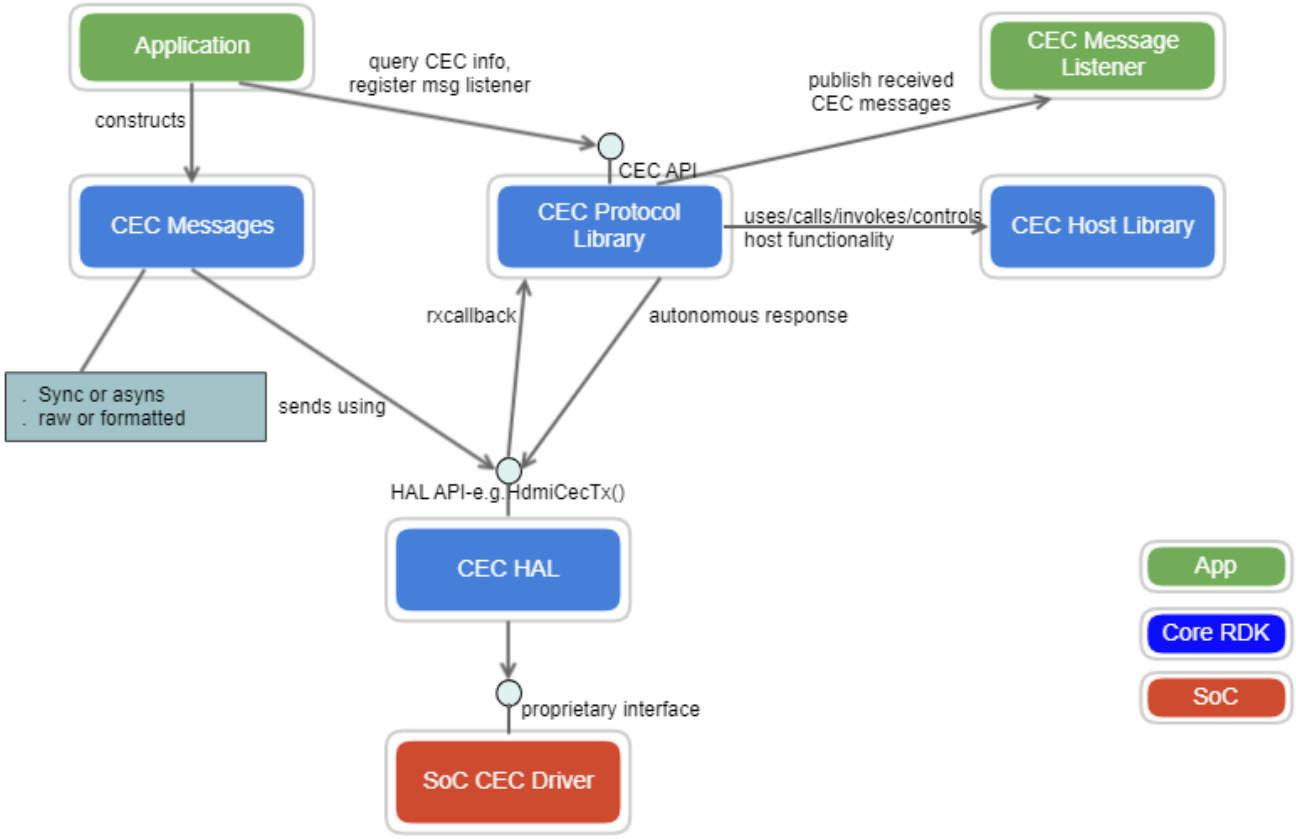
- Hardware support for HDMI-CEC as specified in HDMI 1.4a
    - CEC 4 – Electrical Specification
    - CEC 5 – Signaling and Bit Timings
    - CEC 6 – Frame Description
    - CEC 7 – Reliable Communication Mechanisms
    - CEC 8 – Protocol Extensions
    - CEC 9 – CEC Arbitration
    - CEC 10.1 – Physical Address Discovery
    - CEC 11 – Switch Requirements
- Send raw or formatted CEC messages on behalf of application
- Provide received CEC messages to registered application listener
- Get information about a device connected to HDMI input
- Autonomously handle subset of CEC messages
- Provide list of connected CEC capable device to application
- Send "Feature Abort" response if message is not handled autonomously by RDK and message is not handled by application listener

Device Applicability

This feature applies to the following devices:

- Hybrid
- IP

Architecture Overview



HDMI-CEC Profiles supported in RDK

Profiles	Description
Discovery	Discover HDMI devices that support CEC and provide settop information to those devices
Power	Synchronize settop power state with HDMI device power state
Switching	Switch settop HDMI inputs to settop HDMI outputs
Channel Change	Change channel on settop from HDMI device
Audio	Control audio mute/volume on HDMI device from settop and vice versa
User Input	Accept user input commands from HDMI device

Component Impacts

CEC Protocol Library

- Translate application commands to CEC commands
- Receive CEC commands and provide to application listeners
- Autonomously respond to a subset of CEC messages

HDMI-CEC HAL

- Abstracts SoC HDMI-CEC driver from higher level components that means it will abstract SoC CEC library from CEC Protocol Library
- it allow transmits/receives HDMI-CEC messages

SoC CEC Driver

- Serialize and send CEC commands
- Receive CEC commands and provide to CEC Protocol Library
- Send EDID to HDMI source

- Provide HDMI input connect notification

Service Manager

- Provide HDMI input connect notification via State Observer API.
- Select select video source

Device Settings HAL

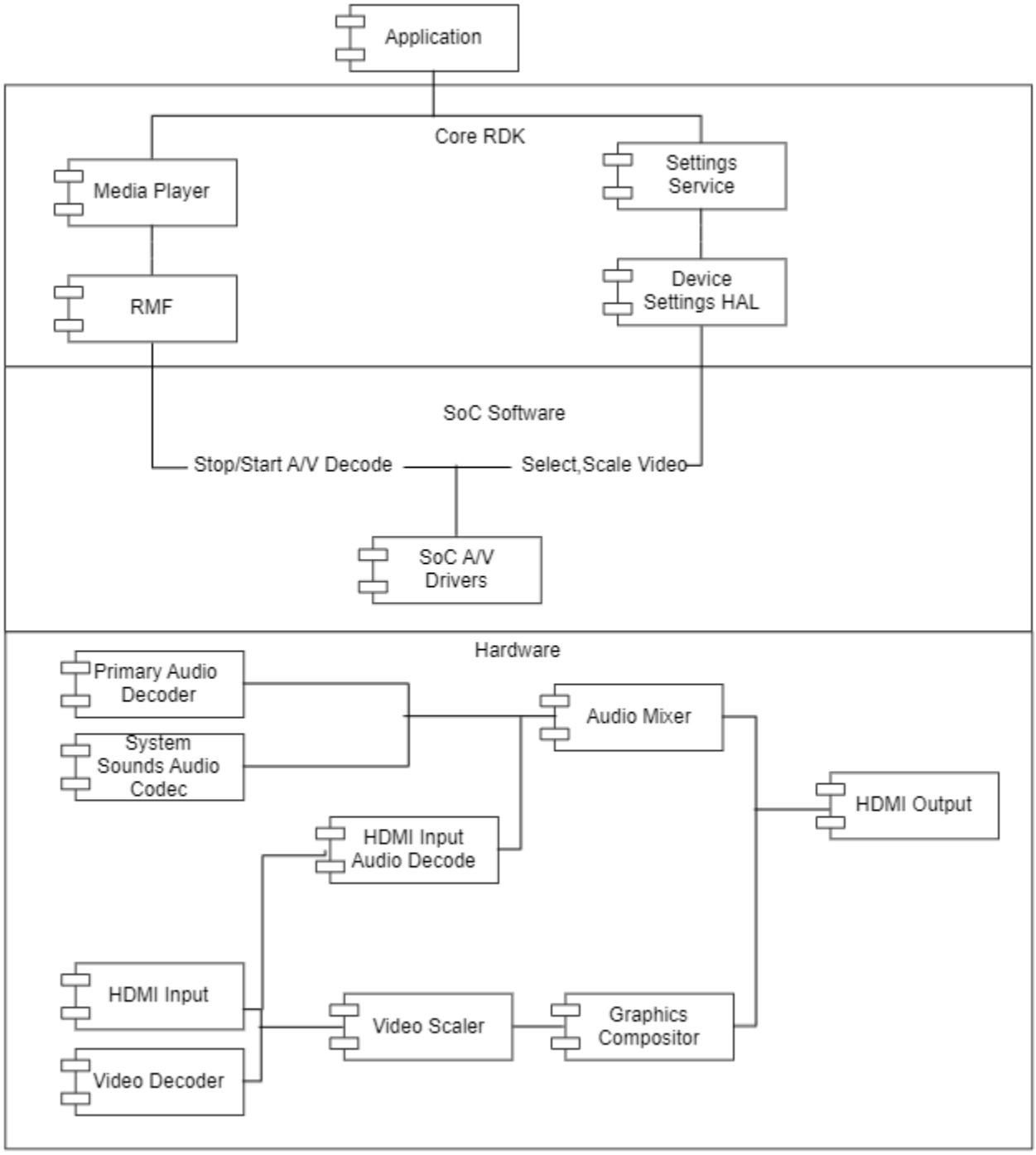
Device Settings APIs that SoC vendors implement. It provides primitive and hardware specific implementation for each controllable aspect of the SoC. This level API is considered single-app mode only, even though its SoC implementation may potentially support multiple-app mode.

- Initialize and terminate device inputs and outputs.
- Determine device settings capabilities (e.g. supported video resolutions, audio modes, etc).
- Modify device settings (e.g. audio encoding).
- Modify front panel LEDs.
- Provide HDMI event notification
- Read/Modify/Write EDID
- Add dsRegisterHdmiListener
- Add dsSelectVideoSource
- Add dsScaleVideoSource

SoC Video Pipeline

- Source video from HDMI input

How HDMI-CEC Works in RDK



Application API Specification

Overall the RDK-CEC library offers 3 categories of application APIs,

HDMI-CEC Connection

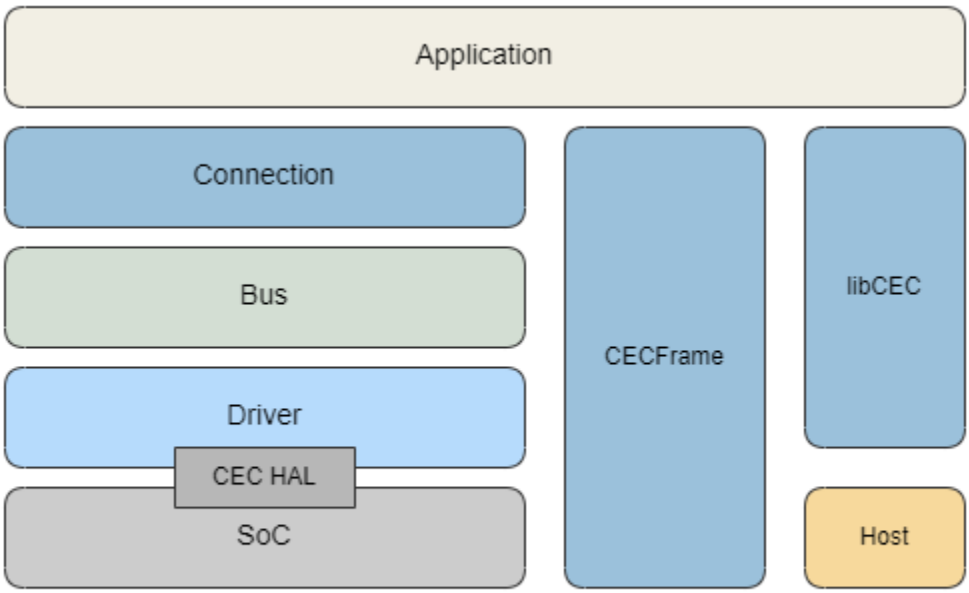
- The Connection APIs allows application to transmit and receive raw data bytes (a.k.a. CEC Frame) that conforms to the HDMI-CEC specification.
- This is the only interface that the application can use to access CEC Bus.

HDMI-CEC Message and Frame Structure

- The Messages APIs allows application to encode high-level message constructs into CECFrame raw bytes, or decode CECFrame raw bytes into high-level message constructs.

HDMI-CEC Library Interface

- The Library APIs allows application to contrl how the CEC stack operates, such as adding logical address to the stack.



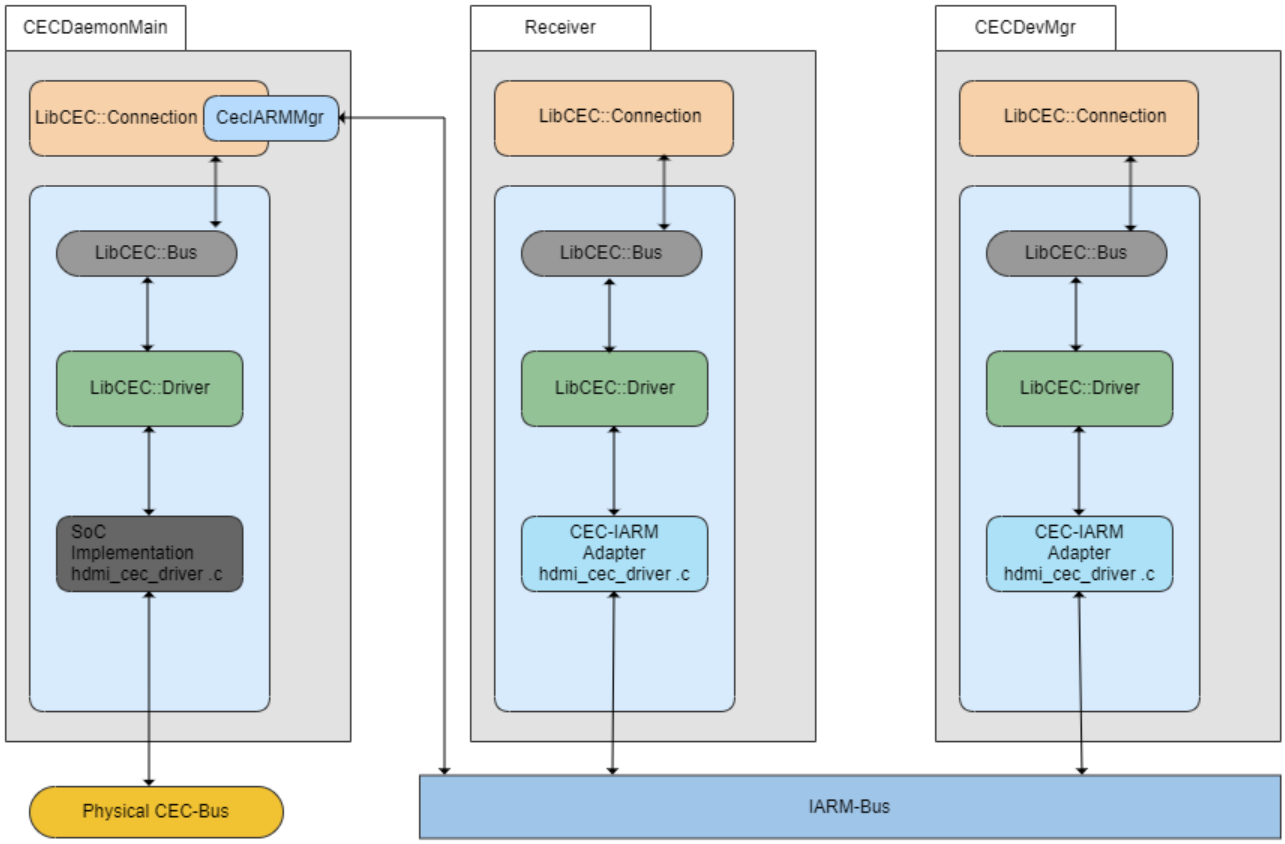
Components of CEC Library

The **Host** interface allows libCEC implementation to interact with the host environment. Such interaction includes monitoring of the Power State change, the HDMI HotPlug events, or API to change the Host State. The Host Interface is delivered as a run-time plugin to the libCEC stack. This allows the CEC stack to run in any devices that implements the Host Interface.

The **Driver** Component access the HDMI-CEC SoC Driver via the CEC HAL API. The vendors are responsible in delivering a SoC Driver that conforms to the HAL API (see the header file hdmi_cec_driver.h)

HDMI-CEC Applications

The relation between Application, Connection and CEC-Bus is described in the figure.



CEC Daemon

- This daemon receives Raw Bytes from the SoC Driver and dispatches them into IARM Bus via CecIARMMgr, it also receives Raw Bytes from IARM Bus and send to the SoC Driver. It controls the only access point to the physical HDMI CEC Bus.

Receiver

- This application receives Raw Bytes from IARM Bus and dispatch to Service Manager (which in turn may dispatch the bytes to Guide Application). It also receives Raw bytes from Service Manager and send to the IARM-Bus.

CECDevMgr

- This application is a "sniffer" on the CEC Bus. It monitors all messages on the bus and construct a Device Map, which depicts the topology of all connected devices on the CEC Bus. Other applications can be developed similar to Receiver or CECDevMgr where the CEC Raw bytes (in form of CECFrame) are send/receive to/from IARM Bus. This pseudo CEC Driver implemented on IARM Bus is called CEC IARM Adapter in the diagram.

HDMI CEC Messaging

The **CECFrame** is a byte buffer that provides access to raw CEC bytes. CECFrame is guaranteed to be a complete CEC Message that has the necessary data blocks:

The **Header** Data block (The byte that contains the initiator and destination address)
The **OpCode** Data Block (The byte that contains the opcode).
The **Operand** Data Block.(The bytes that contains the operands).

In most cases application need not access **CECFrame** directly, but manipulate the raw bytes through the Message API. The Message API allows the application to send or receive high-level CEC message construct instead of raw bytes. Basically for each CEC message (such as ActiveSource), there is a C++ class implementation representing it. Each message class provides necessary getter and setter methods to access the properties of each message.

Asynchronous Vs. Synchronous

When messages converge on the logical buses, they are queued for sending opportunities on the physical bus. The waiting time for such send to complete, though short in most cases, can be problematic to some interactive real-time applications. It is recommended that the applications always send CEC messages asynchronously via the **Connection** API and use the listener APIs to monitor response messages or device state changes. The CEC library offers abundant APIs to facilitate such asynchronous implementation and the application is encouraged to make full use of them.

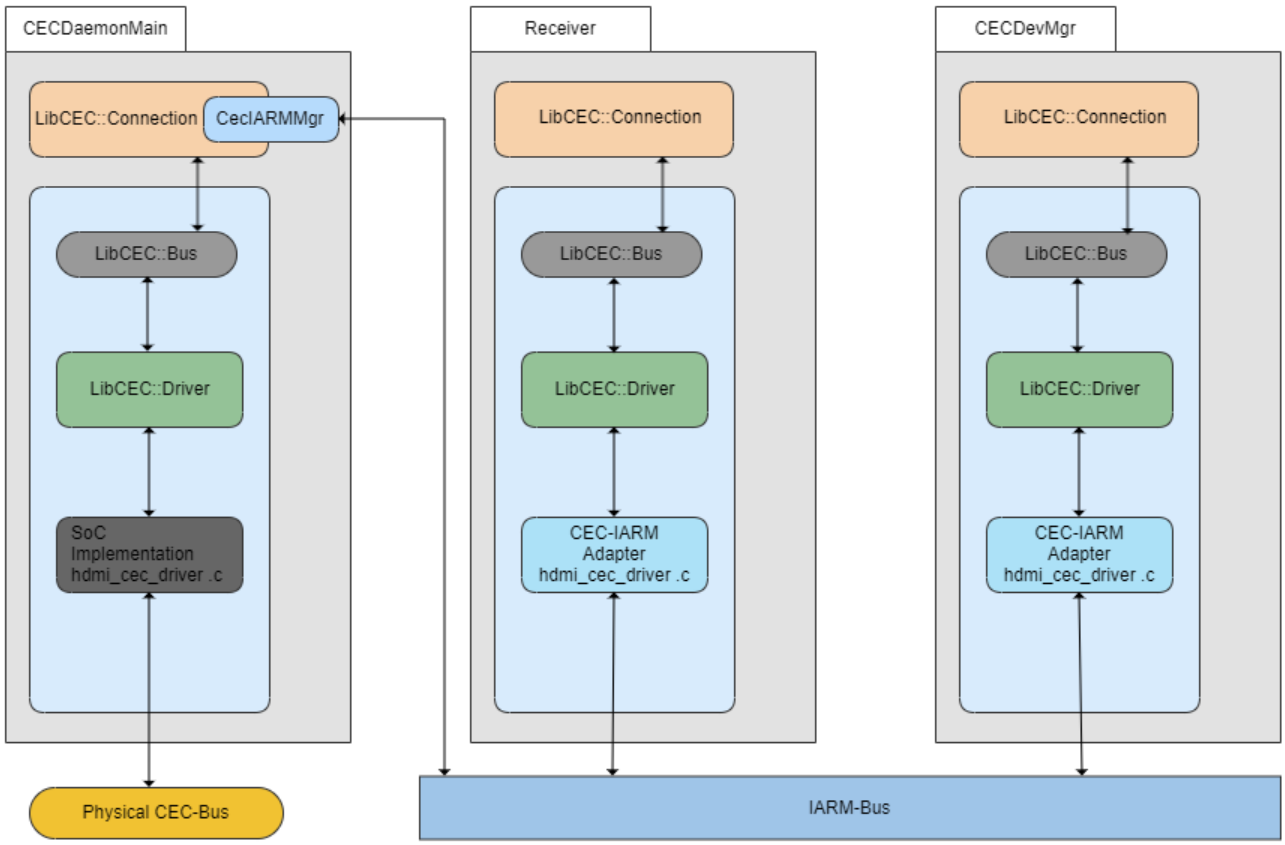
Given the vast variance of HDMI-CEC support from the off-the-self media devices, it is not recommended that application wait for the response from a destination device. Even if the request message is sent out successfully, the destination device may choose to ignore the request. The recommended approach is again to send the request asynchronously and use the listener to monitor responses.

Overall, given the asynchronous nature of HDMI-CEC, application should always opt to use Asynchronous APIs as first choice. And for same reasons, the RDK CEC library offers only limited support for Synchronous APIs.

Multiple Application Support

Often , the application functionality (Record, Tune and Playback) is distributed across multiple components. In order for any component to have equivalent access to the HDMI-CEC bus, the library offers Multi-App support via IARM-Bus. This support is enabled by default, and can be disabled if desired.

In essence, there is only one physical CEC bus on a system. However, with the help of Connection, Logical CEC-Bus, and IARM-Bus, the CEC library can converge the CEC traffic from different Connections and Logical Buses before forwarding them to the single physical bus. This is illustrated by the diagram below.



In this diagram there are two applications (Receiver and CECDevMgr). Since both applications can only access the underlying physical CEC Bus via Connection API, they have no knowledge how the message are eventually delivered to the Physical Bus.

For both applications, its CEC messages flows from,

> Connection --> Logical Bus--> CEC IARM Apaper--> IARM Bus---> (CECDaemonMain)

For CECDaemonMain, its CEC messages flows from,

> IARM Bus --> CecIARMMgr --> Connection --> Logical Bus --> CEC Driver --> (Physical Bus)

The message flow on Connections and Logical Buses are full duplex.

Methods & Events

Following methods are used for DMI CEC module.

Name	Parameters	Description

setEnabled	enabled : boolean	Enables or disables CEC
getEnabled	none	Returns true if CEC is enabled
setName	name : string	Sets the name of the STB device. The default name is "STB". It is recommended that the name of the device is set prior to enabling CEC.
getName	none	Returns the name of the STB device
sendMessage	message : String	The message is a base64 encoded byte array of the raw CEC bytes. The CEC message includes the device ID for the intended destination.

getCECAddresses	none	return the JSON object <*CECAddresses*> that is assigned to the local device. It does not contain the <CECAddresses> of other devices on the connected CEC network.
		
		`"CECAddresses" : {`
		` "physicalAddress": Array of 4 bytes [byte0, byte1, byte2, byte3],`
		` "logicalAddresses" : Array of <CECLogicalAddress>`
		`}`
		
		`"CECLogicalAddress" : {`
		` "deviceType" : <string>`
		` "logicalAddress": <integer>`
		`}`
		
		The *deviceType* returned is part or all of devices types optionally set by XRE.
		
		A CEC device can have multiple deviceTypes, if so an array of <CECLogicalAddress> with size more than one is returned.
		
		Default *deviceType* is Tuner, and its logical Address is either 3, 6, 7, or 10.
		
		In messages sent by XRE, XRE can only use the logicalAddress returned from *CECAddresses*.
		
		Accepted *deviceType* are: "Tuner", "Record", "Playback", "AudioSystem", "VideoProcessor", "Switch"

Events notification:

Name	Content	Description
onMessage	message : String	Fired when a message is sent from an HDMI device. Message is a base64 encoded byte array of the raw CEC bytes.
cecAddressesChanged	JSON object <CECAddresses>	Notify that address of the host CEC device has changed

API Documentation

To know more about SoC/Application level APIs details use in RDK, refer the link RDK HDMI-CEC API Documentation