RDK-V Debugging and Logging

Debugging and logging are both inseparable aspects of any software system during it's development as well as in the deployment stage. These tools greatly assist the developer & support teams to quickly find a solution.

- Debugging tools and methodologies identifies what went wrong thus enabling developer to fix the issue quickly.
- Logging enables to maintain a history of how a software or the piece of code behave in different environments (whether it is in development or in field deployment). It helps in catching nagging field issues which are otherwise impossible to reproduce in a simulated environment. Also the other advantage it provides by generating a solid data set for any analytics process.

Debugging

In the development process of a feature rich software like RDK, debugging the code or modules bears great significance.

An IDE provide great deal by helping to solve certain debug functionalities but when working with a Set-top, it is also important to have debuggers that can collect core dumps and real time code traces.

RDK supports following tools for making the debugging task easier for developers.

- rmfApp
- Breakpad

Logging

Presentation of system events, errors and application traces altogether in form of console messages or persistent log files constitutes the logging process in RDK.

Generating and reading system logs is an important aspect of system administration. The information in system logs can be used to detect hardware and software issues as well as application and system configuration errors. This information also plays an important role in assessing device health, security loopholes and incident response.

There are multiple levels of logging supported by RDK right from having the native Linux syslog (or more recently : journal logs) facility to having a dedicated logging facility such as RDK logger library.

RDK Logger

Description

RDK Logger is responsible for centralizing logging feature which otherwise was done by printf or other similar functionalities. It provides enable/disable log feature as well as defines logging levels thus providing a way to control amount of outgoing log messages. Available as a library, it can be easily plugged into different application to enable them in creating logs in a way that is common among the RDK system.

Capabilities

- Abstracts logging client from underlying logging utility.
- Dynamically enable/disable logging level at run time.
- Provide logging format that complies with the existing format understood across RDK components(e.g. <timestamp> [mod=*, level=*]).

A detailed description on the RDK logger component can be found here: RDK Logger

Systemd Journals

systemd has its own logging system called the journal; therefore, running a syslog daemon is no longer required.

- Journal stores all logs in a specified folder
- journalctl allows you to filter the output as needed
- In order to comply with the RDK logging formats, journalctl is used to redirect logs to specific module level log .txt files
- The future plan is to switch to using journald systems-journal-upload to upload logs