RDK-B Logger

- Introduction
- Architecture
- RDK Logger
- Log4c
- How to add rdklogger to a new component
- RDK Logger build instructions
- API Specification

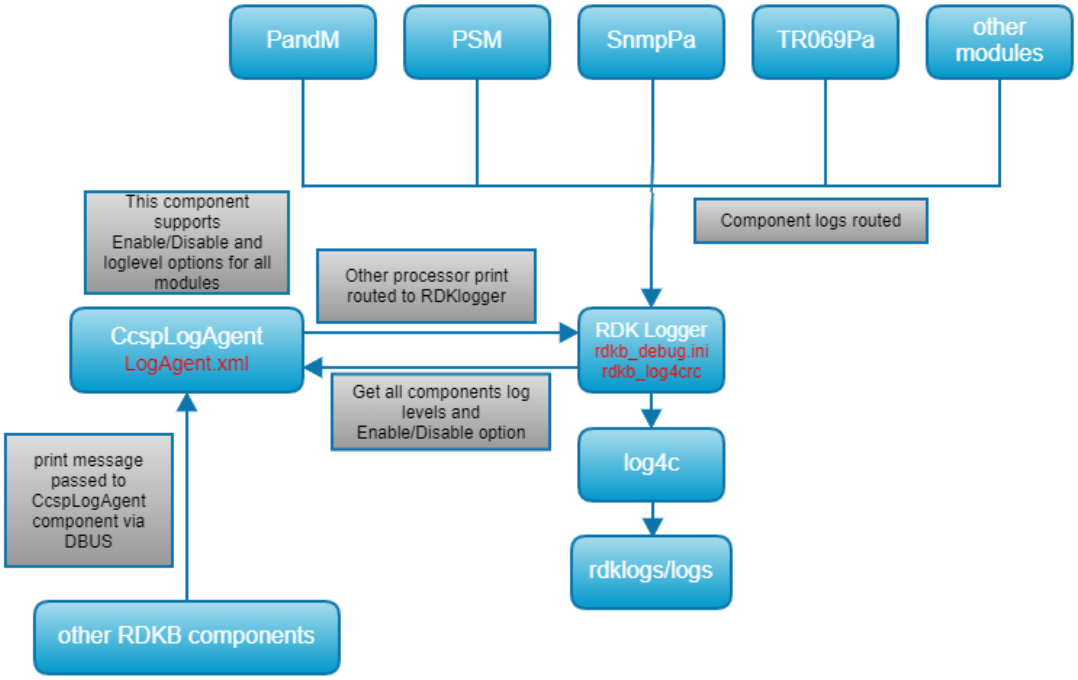
Introduction

RDK logger is a general purpose logging mechanism which is used for logging in RDK-B. Internally this uses log4c for formatting and supporting multiple log levels for different modules. The log level for each component is read from a configuration file debug.ini (rdkb_debug.ini) during component initialisation. Logger is implemented as a shared library and components need to link this library in order to include logging functionality.

RDK logger is linked to RDK-B modules to enable component wise logging. When the unit boots-up, each RDK-B component will be initialised which in turn will initialise logger and start logging into the corresponding log file depending on the log level set for the component.

Note: The RDK data model naming convention prefix was changed in March 2020 to “X_RDK_”. We request you use this new prefix going forward.

Architecture



1. All logs from modules (CcspTraceError, CcspTraceInfo, etc..) are routed to RDK logger library. RDK logger logs these into log files based on module wise ***LogLevels*** & ***LoggerEnable*** options. RDK logger uses log4c library internally to achieve the functionality.

2. CCSPLogAgent:

   All components have separate TR181 parameters to control the LogLevels and LoggerEnable options.

   All components logger TR181 parameters exist in LogAgent.xml file which is handled by CcspLogAgent component. Those parameters are stored in syscfg.db file for persistent storage.

- **LoggerEnable:** This will help to enable/disable logs for particular component. If it is TRUE logs are enabled otherwise logs are disabled.

  Note: We can enable/disable logging for the all components based on “X_RDKCENTRAL-COM_LoggerEnable” parameter. It is by default set to “TRUE”.

- **LogLevels:** We can set different log levels for each component. By default all modules log level is 4 (RDK_LOG_INFO).

  CcspLogAgent component provides the ability to change the log levels at run time for individual components and this component shall be made available to the community shortly.

Below mentioned are the supported LogLevels:

Log Level Index	Log Level
0	RDK_LOG_FATAL
1	RDK_LOG_ERROR
2	RDK_LOG_WARN
3	RDK_LOG_NOTICE
4	RDK_LOG_INFO
5	RDK_LOG_DEBUG
6	RDK_LOG_TRACE1
7	RDK_LOG_TRACE2
8	RDK_LOG_TRACE3
9	RDK_LOG_TRACE4
10	RDK_LOG_TRACE5
11	RDK_LOG_TRACE6
12	RDK_LOG_TRACE7
13	RDK_LOG_TRACE8
14	RDK_LOG_TRACE9

If LogLevel is set to 3, all logs related to 0 to 3 levels are routed to logger which means RDK_LOG_FATAL, RDK_LOG_ERROR, RDK_LOG_WARN and RDK_LOG_NOTICE prints are routed to logger, other prints not routed to logger.

Note: Log level for all components can be controlled using **"X_RDKCENTRAL-COM_LogLevel"** parameter. It is by default set to level 4 which is RDK_LOG_INFO.

Example:

TR069 parametes for RDK logger,

Device.LogAgent.X_RDKCENTRAL-COM_TR69_LoggerEnable

Device.LogAgent.X_RDKCENTRAL-COM_TR69_LogLevel

RDK Logger

RDK logger is dependent on rdkb_debug.ini and rdkb_log4crc configuration files. rkdb_log4crc file is handled by log4c.

rdkb_debug.ini:

- Logs from different components can be logged into separate files based on the "SEPARATE.LOGFILE.SUPPORT" variable.
- It also contains entries for each component log levels. Log levels can be controlled independently for each module.

Example: LOG.RDK.<component name> = ALL, FATAL, ERROR WARNING, NOTICE, INFO, DEBUG

- For the modules not having entries in configuration file, log level is set to default log levels. (Currently this is set to LOG.RDK.DEFAULT = ERROR in debug.ini)
- "rdkb_debug.ini" files are parsed by rdk_logger_init() function for loading all the components entry.

Below function is used for logging in RDKLogger,

RDK_LOG (rdk_LogLevel level, const char *module, const char *format,...)

"level" is log level of the log message (FATAL, ERROR etc}

"module" is Module to which this message belongs to (use module name same as mentioned in debug.ini)

"format" is a printf style string containing the log message.

Log4c

- Log4c is a open source library for flexible logging to files.
- Log4c init “log4c_init()” happens during rdklogger init.
- Log4c loads its configurations from rdkb_log4crc file. This contains the logger information (log directory path, size, file name, etc..).
- LOG4C_RCPATH environment variable holds the path to the log4crc configuration file. So we have to configure this environment variable without fail.
- Currently this is mentioned in cosa_start.sh file (export LOG4C_RCPATH=/rdklogger).

Log4crc File syntax:

The log4crc configuration file uses an XML syntax.

The root element is **<log4c>** and it can be used to control the configuration file version interface with the attribute "version".

The following 4 elements are supported: **<config>, <category>, <appender> and <layout>**.

- The <config> element controls the global log4c configuration. It has 3 sub elements. The <nocleanup> flag inhibits the log4c destructors routines. The <bufsize> element sets the buffer size used to format log4c_logging_event_t objects. If is set to 0, the allocation is dynamic (the <debug> element is currently unused).
- The <category> element has 3 possible attributes: the category "name", the category "priority" and the category "appender". Future versions will handle multiple appenders per category.
- The <appender> element has 3 possible attributes: the appender "name", the appender "type", and the appender "layout".
- The <layout> element has 2 possible attributes: the layout "name" and the layout "type".

Sample log4crc configuration file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE log4c SYSTEM "">

<log4c>

<config>

        <bufsize>0</bufsize>

        <debug level="0"/>

        <nocleanup>0</nocleanup>

</config>

<!-- root category ============================================= -->

<category name="root" priority="notice"/>

<!-- default appenders ==================================== -->

<appender name="stdout" type="stream" layout="basic"/>

<appender name="stderr" type="stream" layout="dated"/>

<appender name="syslog" type="syslog" layout="basic"/>

<!-- default layouts ======================================== --> <layout name="basic" type="basic"/>

<layout name="dated" type="dated"/>

</log4c>
```

Logs will be logged into directory /rdklogs/logs/ for each module by log4c. /rdklogs/logs/ will contain latest logs for each module.

Log file name will be of the format : <modulename>log.txt.<0,1>

Log file for a particular module can be easily identified from the name of the log file. **Example:** TR69log.txt.0

Logs folder size will be periodically monitored (/rdklogs/logs folder) for upload.

rdklogs/logs will be periodically checked for the maximum size set for the logs to be uploaded.(“rdkbLogMonitor.sh” script file monitor this functionality)

In RDK-B maximum size of log directory is set to 1.5MB. This value is chosen based on storage constraints for the target platform. Can be changed to any value based on storage availability.

/rdklogs/logs will contain entire logs from boot-up till the first time upload.

Once the folder size (/rdklogs/logs/) reaches 1.5 MB, then these logs will be moved to /nvram/logbackup folder and uploaded to log server.

In /nvram/logbackup folder, before uploading the logs a folder with current timestamp will be created.

All log files will be moved to /nvram/logbackup/<timestamp> directory.

Then the directory will be compressed and renamed as mentioned below to enable server to fetch the logs based on MAC address of unit : <WAN Interface mac>_LOGS_<Timestamp>.tgz

When the logs folder size reaches 1.5 MB for the second time, logs which are backed up in /nvram/logbackup will be removed.

Typically before rebooting the device, logs will be uploaded to the log server.

Logging from components based on other processor(ATOM):

Logs from components based on other cores (ATOM) are routed to rdklogger via CcspLogAgent component.

ATOM processor components log messages are passed as a parameter to CcspLogAgent component via D-bus. Then CcspLogAgent routs the logs to rdklogger.
Example: In some of the devices WifiAgent component runs on the ATOM processor

How to add rdklogger to a new component

Steps to add rdklogger to a new component:

Step 1:

Need to add LoggerEnable and LogLevel parameters in “LogAgent.xml” file for new component.

```
X_RDKCENTRAL-COM_New_LoggerEnable
X_RDKCENTRAL-COM_New_LogLevel
```

Step 2:

Need to add the new parameters in “system_defaults_arm” file for persistent storage.

```
$ X_RDKCENTRAL-COM_New_LogLevel=4
$ X_RDKCENTRAL-COM_New_LoggerEnable=1
```

Code functionality for Loglevel & LoggerEnable parameters set/get in “cosa_apis_logagentplugin.c” file

Initialise rdklogger by calling rdk_logger_init (/fss/gw/lib/debug.ini) in the new component

Step 3:

Add new component entry in “rdkb_debug.ini” file

```
LOG.RDK.New = ALL FATAL ERROR WARNING NOTICE INFO DEBUG TRACE
```

Step 4:

Add the below configuration in “rdkb_log4src” file for new component

```
<rollingpolicy name="New_rollingpolicy" type="sizewin" maxsize="2097152" maxnum="2"/>

<appender name="RI_Newrollingfileappender" type="rollingfile" logdir="/rdklogs/logs/" prefix="Newlog.txt" layout="comcast_dated" rollingpolicy="
New_rollingpolicy"/>

<category name="RI.Stack.New" priority="debug" appender= "RI_Newrollingfileappender"/>

<category name="RI.Stack.LOG.RDK.New" priority="debug" appender= "RI_Newrollingfileappender"/>
```

RDK Logger build instructions

RDK Logger is available in the below path

../source_code/components/generic/rdk_logger

Below bitbake commands are used for building rdk-logger

```
$ bitbake -c compile -f rdk-logger

$ bitbake rdk-logger

$ bitbake rdk-generic-broadband-image
```

API Specification

For Doxygen documentation on the RDK Logger please refer: Doxygen RDK Logger