VCS21 How To Enable SIP Stack Debug

Table of Contents

[1.0 Introduction 4](#_Toc374452894)

[1.1 Overview 4](#_Toc374452895)

[2.0 How To 4](#_Toc374452896)

[2.1 Enabling Log level in the SIP Lib 4](#_Toc374452897)

[2.2 Enable Logging within application 4](#_Toc374452898)

[3.0 Build / run 5](#_Toc374452899)

[3.1 SIP Library 5](#_Toc374452900)

[3.2 Application 5](#_Toc374452901)

[4.0 Notes: 5](#_Toc374452902)

[5.0 References 7](#_Toc374452903)

Revision Record

| Rev | Date | Author | Comments |
| --- | --- | --- | --- |
| 0 | 12/10/2013 | Drew Orr | Initial Creation |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Overview

The purpose of this Document is to show the user how to enable logging in an application using the sip stack library. Changes to both the application and the sip library are required and each will have to be built and deployed to the target machine.

# How To

## Enabling Log level in the SIP Lib

You will first need to select the log level you wish to display. Identify the specific statements you want to output and take note of the trace level. (Example below)

**MX\_TRACE6(0, g\_stSipStackSipCoreCSipStackMonitor,**

**"CSipStackMonitor(%p)::CSipStackMonitor()", this);**

You can then use any number of the corresponding level defines or the catch all defines. This list can be found in both the api refrence document or within the following header.

[third\_party/sip/m57-sip-4.1.8.13/M5TFramework\_v2\_1/Sources/Basic/MxTrace.h](https://lnsvr0011/repositories/ion/embedded/ccu/base/trunk/third_party/sip/m5t-sip-4.1.8.13/M5TFramework_v2_1/Sources/Basic/MxTrace.h)

• MXD\_TRACE0\_ENABLE\_SUPPORT

• MXD\_TRACE1\_ENABLE\_SUPPORT

• MXD\_TRACE2\_ENABLE\_SUPPORT

• MXD\_TRACE3\_ENABLE\_SUPPORT

• MXD\_TRACE4\_ENABLE\_SUPPORT

• MXD\_TRACE5\_ENABLE\_SUPPORT

• MXD\_TRACE6\_ENABLE\_SUPPORT

• MXD\_TRACE7\_ENABLE\_SUPPORT

• MXD\_TRACE8\_ENABLE\_SUPPORT

• MXD\_TRACE9\_ENABLE\_SUPPORT

• MXD\_TRACEX\_ENABLE\_SUPPORT

• MXD\_TRACE\_ENABLE\_ALL

• MXD\_TRACE\_MAX\_LEVEL

Once one or more levels are selected, you will need to add them to the PreMxConfig.h file to enable them.

[third\_party/sip/m57-sip-4.1.8.13/Build/Sources/PreMxConfig.h](https://lnsvr0011/repositories/ion/embedded/ccu/base/trunk/third_party/sip/m5t-sip-4.1.8.13/Build/Sources/PreMxConfig.h)

for example: #define MXD\_TRACE\_ENABLE\_ALL

## Enable Logging within application

The second step in setting up the SIP logs is to enable the specific nodes you wish to see logging from and to enable them in your application (MAP, TMG, etc..)

Navigate to the following file where we will add the enable statements around line number 320.

[Service/common/src/atcoip/SipUA/SipUA.cpp](https://lnsvr0011/repositories/ion/embedded/ccu/base/trunk/service/common/src/atcoip/SipUA/SipUA.cpp)

You may find some statements already there enabling nodes, I like to comment those out and start fresh so I get only what I expect. After commenting them out, first line I like to add is the following, It will ensure no other nodes will be enabled at this point forward.

**MxTraceDisableNode("/", true);**

Next you will need to select The nodes you wish to enable. To do this, you locate the trace statements you wish to see and take not of the second parameter. This is the node definition. Lets take the log statement example from above which uses the following node **g\_stSipStackSipCoreCSipStackMonitor** Add the following line to enable logging for this node

**MxTraceEnableNode("/SipStack/SipCore/CSipStackMonitor", true);**

The above statement is probably a bit confusing, how did we get from the node name to a path string? Fist start off by doing a search for the node name within the code base, we want to find the location where it is registered ( a lot of times it is found in **CSipStackInitializer.cpp**). Once we find the registration, we can trace it backwards to create our string. I will show the steps below.

* Find the registration

**MxTraceRegisterNode(&g\_stSipStackSipCore,**

**&g\_stSipStackSipCoreCSipStackMonitor,**

**"CSipStackMonitor");**

* Now we trace backwards using the first parameter for our next registration search

**MxTraceRegisterNode(&g\_stSipStack, &g\_stSipStackSipCore, "SipCore");**

* Repeat until we get to the root

**MxTraceRegisterNode(&g\_stTraceRoot, &g\_stSipStack, "SipStack");**

# Build / run

## SIP Library

To build the SIP library, execute the following build script using correct target

**build/core/sharedlibs\_build.sh sip arm**

Now deploy the libM5tSipUa.so under the /home/lib/ directory on the target machine

## Application

Build your application and deploy to target machine. Start your process manually through the command line, as the sip logging will be printed to screen.

# Notes:

* For anyone trying to debug shutdown routines, I have found that about 25% through the shutdown sequence, the library unregisters all its nodes, thus turning off logging. Even though they have TRACE statements throughout their Finalize() routines, you will not see them once the Un-Registration of the nodes occurs.

# References

Table 2.0-1 lists the documents cited herein.

Table 2.0-1 Referenced Documents

| Title | Location |
| --- | --- |
| M5T Framework API | [/embedded/ccu/base/trunk/third\_party/sip/m5t-sip-4.1.8.13/M5TFramework\_v2\_1/Documentation](https://lnsvr0011/repositories/ion/embedded/ccu/base/trunk/third_party/sip/m5t-sip-4.1.8.13/M5TFramework_v2_1/Documentation/M5T%20Framework%20v2.1%20-%20API%20Reference.pdf) |