User Guide for Logging

This is guide is just a simple summary about Log Service, invoking log APIs, checking log files and changing configuration parameters. For detail information, please find in the following document:

http://upload.wikimedia.org/wikipedia/mediawiki/4/4f/SAI-AIS-LOG-A.02.01.pdf

Log Service

Log Service is one function integrated in coremw. Logging information is a high-level cluster-significant, function-based (as opposed to implementation-particular) information suited for us, or automated tools to review current and historical logged information to trouble shoot issues such as misconfigurations, network disconnects and unavailable resources.

Log Stream

A log stream is a conceptual flow of log records. The Log Service enables applications to express and forward log records through well-known log streams that lead to particular output destinations such as a named file.

There are four kind of log stream(alarm,system,notification,application). There is exactly one log stream for each of the alarm, notification, and system log stream types, However, there can be any number of application log streams.

Log API

Function Name	Description
saLogInitialize()	This function initializes the Log Service for the invoking process and registers the various callback functions. This function must be invoked prior to the invocation of any other Log Service functionality
saLogSelectionObjectGet()	This function returns the operating system handle, selectionObject, associated with the handle logHandle. The operating system handle returned by saLogSelectionObjectGet() is valid until saLogFinalize() is invoked on the same handle logHandle.
saLogStreamOpen_2() saLogStreamOpenAsync_2()	This function open a stream and return log stream handle. For the three well-known log streams, the returned log stream handle refers to the existing alarm, notification, or system log streams, which are created when the Log Service is initialized in the cluster. These log streams persist over the lifetime of the Log Service in the cluster.
saLogWriteLog() and saLogWriteLogAsync()	These API functions are used to log a record. Writing a log record to a log file is an atomic operation, so that concurrent writes must be properly handled.
saLogStreamClose()	The invocation of this API function closes the log stream which was opened by an earlier invocation of saLogStreamOpen_2() or saLogStreamOpenAsync_2(). This call frees all resources allocated for this process by the Log Service on the log stream. If the invocation of the saLogStreamClose() function completes successfully, and the log stream is an application log stream, and no other process has that application log stream open, the Log Service behaves as follows:

	The log stream is deleted. The log file associated with that application log stream is closed and renamed with a <closetime> that indicates when the last user of the log stream closed the stream. The log file configuration file associated with the deleted log stream is closed and persists indefinitely.
saLogFinalize()	The saLogFinalize() function closes the association between the invoking process and the Log Service. The process must have invoked saLogInitialize() before it invokes this function. If the saLogFinalize() function completes successfully, it releases all resources acquired when saLogInitialize() was called. If a process terminates, the Log Service implicitly finalizes all instances of the Log Service that are associated with the process.
SaLogStreamOpenCallbackT SaLogWriteLogCallbackT SaLogFilterSetCallbackT saLogDispatch()	Please refer to the up attachment.

Imm Command Usage

- immfind | grep Log

we could check all the existing objects with this command and we can get Log stream object domain as follows:

```
sh-3.2# immfind | grep Log
opensafGlobalLogPolicy=safDtsv,safApp=safDtsvService
safApp=safLogService
safLgStrCfg=saLogAlarm,safApp=safLogService
safLgStrCfg=saLogNotification,safApp=safLogService
safLgStrCfg=saLogSystem,safApp=safLogService
```

- immlist

For example: immlist safLgStrCfg=saLogAlarm,safApp=safLogService
We can use this command to check configuration parameters of each log stream.

```
sh-3.2# immlist safLgStrCfg=saLogAlarm,safApp=safLogService
Name                                            Type         Value(s)
========================================================================
safLgStrCfg                                     SA_STRING_T  safLgStrCfg=saLogAlarm
saLogStreamSeverityFilter                       SA_UINT32_T  127 (0x7f)
saLogStreamPathName                             SA_STRING_T  .
saLogStreamNumOpeners                           SA_UINT32_T  2 (0x2)
saLogStreamMaxLogFileSize                       SA_UINT64_T  5000000 (0x4c4b40)
saLogStreamMaxFilesRotated                      SA_UINT32_T  4 (0x4)
saLogStreamLogFullHaltThreshold                 SA_UINT32_T  75 (0x4b)
saLogStreamLogFullAction                        SA_UINT32_T  3 (0x3)
saLogStreamLogFileFormat                        SA_STRING_T  @Cr @Ct @Nt @Ne6 @No30 @Ng30 "@Cb"
saLogStreamFixedLogRecordSize                   SA_UINT32_T  200 (0xc8)
saLogStreamFileName                             SA_STRING_T  saLogAlarm
saLogStreamCreationTimestamp                    SA_TIME_T    9493851860000000000 (0xd2ce4cbee8c5400, Tue Feb  1 06:06:26 2000)
SaImmAttrImplementerName                        SA_STRING_T  safLogService
SaImmAttrClassName                              SA_STRING_T  SaLogStreamConfig
SaImmAttrAdminOwnerName                         SA_STRING_T  IMMLOADER
```

- immcfg

We take “saLogStreamSeverityFilter” as example.
The parameter “saLogStreamSeverityFilter” ranges from 0 to 127.
Each bit stands for a type of severity and from low to high is:

	emerg	alert	crit	error	warn	notice	info

In each bit, 1 stands for “enable” while 0 stands for “disable”. The log stream of different severity type could be created only when the corresponding bit is set to 1. For example:
immcfg –a saLogStreamSeverityFilter=7 safLgStrCfg=saLogSystem,safApp=safLogService

The value turns to 0000111, only low three bits are enabled, which means only log stream of **emerg**, **alert** and **crit** level log could be created, the others are filtered out.

Note:

- The parameter “saLogStreamSeverityFilter” of Alarm and Notificaion stream can not be configured, it must be 127.
- Just configuration parameter of System Stream can be changed by immcfg command.
- For Application Stream, we have to use another immadm command because application stream is runtime object. If the application stream object is exsting, we could modify its parameter. When the process has invoken the API saLogStreamClose() and , it means that this application stream has been deleted, so we can’t change parameters any more.

- immadm

For example:
immadm -o 1 -p saLogStreamSeverityFilter:SA_UINT32_T:15 safLgStr=App_name

This command is used to configure saLogStreamSeverityFilter value of Application log. The saLogStreamSeverityFilter value function of each bit is the same as the three other streams.

Demo Command

saflogger is an executable command by opensaf, we use it to verify the logging function simply.

saflogger [options] [message ...]

OPTIONS

-l or --alarm	write to alarm stream
-n or --notification	write to notification stream
-y or --system	write to system stream (default)
-a NAME or ---application=NAME	write to application stream NAME
-s SEV or --severity=SEV	use severity SEV, default info valid severity names: emerg, alert,crit, error, warn, notice, info

-i INT or --interval=INT write with interval INT (only with --count, default 1s)

-c CNT or --count=CNT write CNT number of times, -1 forever (with interval INT)

For example:

saflogger --alarm --severity=emerg

the log file name format is saLogAlarm_system time.log

saflogger --application=App-name --severity=emerg

A new Application log is created when a new Application log stream is written, and the log file name format is Appname_start-time_end-time.log