# Logging Guidelines

## Log Formats

Log files should be written so that each log entry follows these rules:

- Log entries are only text data. UTF-8 is allowed, ASCII is preferred, binary data is forbidden.
- **Never** log any security-critical values (passwords, keys, system secrets)
- An entry is a single line of text.
- The beginning of the entry must be 4 fields, each terminated by a colon ':' character:
  1. An ISO format date/time stamp to six decimal places inside double quote characters. If the available time precision is less than six decimal places, pad with zeros. The timestamp **must** be in Universal Time (UTC) *not* local system time.

     **"***yyyy-mm-dd***T***hh***:***MM***:***ss***.***ffffff***Z":**

  2. A decimal sequence number (variable length, no left padding with zeros required)
  3. A facility name (values are arbitrary, but should be chosen to be useful for filtering)
  4. The log level, using the keywords from #Log Levels below.
- The remainder of the line may be any format, but Newline characters, backslash, and double quotes in the log text must be escaped by preceding them with a backslash.

## Log Levels

The levels to be used are described in the following table, in order of decreasing verbosity and increasing severity.

| Level | Description |
|---|---|
| **DEBUG** | Developer messages needed only when debugging code. May include recording of such things as entry and exit from methods or internal branch tracking.<br>***This level should never be required by end users.*** |
| **INFO** | Informational message used to trace system inputs and actions. Significant actions in the execution of some activity;<br>for example, the receipt of a message or an important decision in its disposition.<br>**This level should be sufficient for an administrator or support person to determine what occured in the system when debugging a configuration or interoperability problem in the field.** |
| **NOTICE** | Normal but significant events. Events that are expected but provide important context, such as service restarts and reloading configuration files.<br>This is the default logging level, and generally logging should always include at least these messages. |
| **WARNING** | Conditions that imply that some failure is possible, but not certain. Generally, external inputs that are not as expected and possibly invalid. Especially useful in low level routines that are going to return an error that may be recoverable by the caller. |
| **ERR** | An unexpected condition likely to cause an end-user visible failure.<br>**This level should be used whenever an error response is being sent outside the system to provide a record of the internal data that are important to understanding it.** |
| **CRIT** | Endangers service operation beyond the current operation.<br>***MUST be logged prior to any 'assert', or when exiting for any abnormal reason.*** |
| **ALERT** | Fault to be communicated to operations.<br>*Should be replaced by usage of the Alarm subsystem.* |
| **EMERG** | *Should be replaced with CRIT.* |

## Libraries

Code written for the sipXecs project should use logging libraries from the project: the `OsSysLog` class (in `sipXportLib`) for C++. Java code should use the log4j logging framework and use the log4j Appender defined in *`org.sipfoundry.commons.log4j.SipFoundryAppender`* and the log4j Layout defined in *`org.sipfoundry.commons.log4j.SipFoundryLayout`*.

*Where possible, third party code should have its logging wrapped to conform to these conventions.*