ASL Documentation

Release 1.1

Ronald Oussoren

Jan 14, 2020

Contents

This package implements bindings to the ASL library on macOS 10.4 or later. The ASL library is a low-level logging library, meant as a replacement for the syslog API.

The library supports Python 3.6 or later.

Note: The ASL library itself is deprecated on macOS 10.12 and is replaced by the os_log library.

Contents:

CHAPTER 1

asl — ASL library

The Apple System Log facility is a logging API with a simular goal to the classic unix syslog API, but with a richer interface and with an API for querying the logging system.

There is more information on the ASL library in Apple's manual page for ASL

Note: All APIs accept unicode strings, and require them for Python 3.

The strings will be converted to the C strings expected by the C API using the UTF-8 encoding.

Note: Functions and methods raise `OSError` when the C API fails.

1.1 ASL connection

class asl.**aslclient**(*ident*, *facility*, *options*)

Parameters

- **ident** – Name of the sender
- **facility** – A facility name
- **optons** – Option flags, see *Open options*

Connection to the ASL backend. These objects implement the context protocol and can be used with the "with" statement.

add_log_file(*fd*)

Parameters **fd** – A file descriptor

Log messages will be written to the file descriptor as well as to the logging subsystem.

remove_log_file(*fd*)

Parameters **fd** – A file descriptor

No longer write log messages to the file descriptor. The file descriptor should be one that's added earlier with *add_log_file()*.

set_filter(*filter*)

Parameters **filter** – an integer with the levels that should be sent to the server

filter is an bitwise or of values from *Message priority levels*, and only messages whose log level is set in *filter* will be sent to the logging subsystem.

Returns the previous value of the filter.

log(*msg*, *level*, *text*)

Parameters

- **msg** – an *aslmsg* object or None
- **level** – a lot level from *Message priority levels*
- **text** – a message string for the log message

Note: The C API uses a printf-style format string instead of a message. In Python you can use string formatting to format the string.

---

send(*msg*)

Parameters **msg** – an *aslmsg* object

Send a log message to the logging subsystem.

search(*msg*)

Parameters **msg** – an *aslmsg* object

Returns an iterator that yields the result messages.

Send a query message to the logging subsystem.

log_descriptor(*msg*, *level*, *fd*, *fd_type*)

Parameters

- **msg** – an *aslmsg* object or None
- **level** – a log level from *Message priority levels*
- **fd** – a file descriptor
- **fd_type** – type of file descriptor, from *File descriptor types*

If *fd_type* is *ASL_LOG_DESCRIPTOR_READ* ASL will read lines from the file descriptor and forward those lines to the logging subsystem as log messages.

If *fd_type* is *ASL_LOG_DESCRIPTOR_WRITE* the file descriptor is closed and reopened as a pipe where the application can write lines that will be converted to log messages.

The *msg* is a template for the log messages created by this API.

This method is available on OSX 10.8 or later.

close()
Explicitly close the client.

Note: The connection will also de closed when the object is garbage collected.

asl.**asl_open**(*ident*, *facility*, *options*)

Parameters

- **ident** – A program identifier string, or None.
- **facility** – A facility name
- **options** – Option flags, see *Open options*

This is an alias for asclient.

asl.**open_from_file**(*fd*, *ident*, *facility*)

Parameters

- **fd** – A file descriptor, open for writing and reading
- **ident** – A program identifier string, or None.
- **facility** – A facility name

Opens an ASL log file for writing using an existing file descriptor, for example one returned by *create_auxiliary_file()*. The file descriptor must be open for reading and writing.

Avaliable on Mac OS X 10.7 or later.

1.2 ASL messages

class asl.**aslmsg**(*type*)

__getitem__(*key*)

Parameters **key** – An attribute name

Return the attribute value for *key*. The key is a unicode string.

See *Standard message attributes* for a list of standard attributes.

__setitem__(*key*, *value*)

Parameters

- **key** – An attribute name
- **value** – Value for the attribute, must be a string

Set the value for attribute *key* to *value*. Both arguments are unicode strings.

See *Standard message attributes* for a list of standard attributes.

__delitem__(*key*)

Parameters **key** – An attribute name

Remove an attribute from the message.

set_query(*key*, *value*, *operation*)

Parameters

- **key** – An attribute name

- **value** – Value to compare the attribute name with
- **operation** – The comparison method

Add a query element to the message. The operation is ... .

A second call to *set_query()* for the same *key* will replace that query. Calls to *set_query()* for different values of *key* are combined into an AND query (that is, all query elements must match).

Note: It is not possible to perform OR queries, to do those you'll have to fetch and merge the various subsets yourself.

Note: For basic equality tests (*ASL_QUERY_OP_EQUAL*) you can also set the *key* and *value* using the mapping interface. That is,

```
m[key] = value
```

is equivalent to:

```
m.set_query(key, value, ASL_QUERY_OP_EQUAL)
```

keys()
Returns the set of attribute names for this message.

asdict()
Return a dict with all attributes of this message. Equivalent to:

```
{ k: msg[k] for k in msg.keys() }
```

Note: It is not possible to retrieve the "operation" for query messages, the C API doesn't provide this information.

1.3 Utility functions

asl.**ASL_FILTER_MASK**(*level*)

Parameters **level** – A message priority level

Converts one of the values from *Message priority levels* into a bit mask that can be used with *aslclient.set_filter()*.

asl.**ASL_FILTER_MASK_UPTO**(*level*)

Parameters **level** – A message priority level

Returns a mask where all bits from *ASL_LEVEL_DEBUG* upto *level* are set.

asl.**create_auxiliary_file**(*msg*, *title*, *uti*)

Parameters

- **msg** – An *aslmsg* object
- **title** – Title for the auxiliary file (for display in Console.app)

- **uti** – UTI for the file format, or None

Creates an auxiliary file that may be used to store arbitrary data associated with the mssage. Returns a file descriptor for the file. This file descriptor must be closed with *close_auxiliary_file()*.

When *uti* is None the system will use "public.data" instead.

The Console.app application will show auxiliary file as an file icon that can be opened.

This function is available on Mac OS X 10.7 or later.

asl.**log_auxiliary_location**(*msg*, *title*, *uti*, *url*)

Parameters

- **msg** – An *aslmsg* object
- **title** – Title for the auxiliary file (for display in Console.app)
- **uti** – UTI for the file format of the URL contents, or None
- **url** – String representation of an URL

Write a log message to the logging system with a URL in the message.

When *uti* is None the system will use "public.data" instead.

The Console.app application will show the URL as a clickable link.

This method is available on Mac OS X 10.7 or later.

asl.**close_auxiliary_file**(*fd*)

Parameters **fd** – File descriptor returned by *create_auxiliary_file()*.

Close the file descriptor for an auxiliary file that was created earlier with `aslmsg.create_auxiliary_file()`. A side effect of this is that the message is logged with the logging system.

asl.**asl_new**(*type*)
This is an alias for *aslmsg*

1.4 Constants

1.4.1 Message priority levels

The levels are listed from highest to lowest priority.

asl.**ASL_LEVEL_EMERG**

asl.**ASL_LEVEL_ALERT**

asl.**ASL_LEVEL_CRIT**

asl.**ASL_LEVEL_ERR**

asl.**ASL_LEVEL_WARNING**

asl.**ASL_LEVEL_NOTICE**

asl.**ASL_LEVEL_INFO**

asl.**ASL_LEVEL_DEBUG**

1.4.2 Message priority level strings

These are the string representation of the constants in the previous section, and are used as the value for the *ASL_KEY_LEVEL* key in *aslmsg* objects.

asl.**ASL_STRING_EMERG**

asl.**ASL_STRING_ALERT**

asl.**ASL_STRING_CRIT**

asl.**ASL_STRING_ERR**

asl.**ASL_STRING_WARNING**

asl.**ASL_STRING_NOTICE**

asl.**ASL_STRING_INFO**

asl.**ASL_STRING_DEBUG**

1.4.3 Priority translations

asl.**LEVEL2STRING**
A directionary mapping numeric levels to the equivalent string value

asl.**STRING2LEVEL**
A directionary mapping string levels to the equivalent integer value

1.4.4 Attribute matching operations

Modifiers

asl.**ASL_QUERY_OP_CASEFOLD**
String comparisons are case folded

asl.**ASL_QUERY_OP_PREFIX**
The match is done on a leading substring

asl.**ASL_QUERY_OP_SUFFIX**
The match is done on a trailing substring

asl.**ASL_QUERY_OP_SUBSTRING**
Match any substring

asl.**ASL_QUERY_OP_NUMERIC**
Perform the comparison after converting the value to an integer using the C function `atoi`.

Operators

asl.**ASL_QUERY_OP_REGEX**
Perform a regular expression match using the regex library. When the `ALS_QUERY_OP_CASEFOLD` modifier is specified the regular expression is compiled case insensitive (*REG_ICASE*). All other modifiers are ignored.

asl.**ASL_QUERY_OP_EQUAL**
Value equality

asl.**ASL_QUERY_OP_GREATER**
Value greater than

asl.**ASL_QUERY_OP_GREATER_EQUAL**
Value greater than or equal to

asl.**ASL_QUERY_OP_LESS**
Value less than

asl.**ASL_QUERY_OP_LESS_EQUAL**
Value less than or equal to

asl.**ASL_QUERY_OP_NOT_EQUAL**
Value not equal

asl.**ASL_QUERY_OP_TRUE**
Always true. Use this to test if an attribute is present.

1.4.5 Standard message attributes

These are the names of well-known attributes of ASL messages, you can add other attributes as well but those won't be used by the ASL backend.

asl.**ASL_KEY_TIME**
Timestamp. Set automatically

asl.**ASL_KEY_TIME_NSEC**
Nanosecond time.

asl.**ASL_KEY_HOST**
Sender's address (set by the server).

asl.**ASL_KEY_SENDER**
Sender's identification string. Default is process name.

asl.**ASL_KEY_FACILITY**
Sender's facility. Default is "user".

asl.**ASL_KEY_PID**
Sending process ID encoded as a string. Set automatically.

asl.**ASL_KEY_UID**
UID that sent the log message (set by the server).

asl.**ASL_KEY_GID**
GID that sent the log message (set by the server).

asl.**ASL_KEY_LEVEL**
Log level number encoded as a string. See levels above.

asl.**ASL_KEY_MSG**
Message text.

asl.**ASL_KEY_READ_UID**
User read access (-1 is any user).

asl.**ASL_KEY_READ_GID**
Group read access (-1 is any group).

asl.**ASL_KEY_EXPIRE_TIME**
Expiration time for messages with long TTL.

asl.**ASL_KEY_MSG_ID**
64-bit message ID number (set by the server).

asl.**ASL_KEY_SESSION**
Session (set by the launchd).

asl.**ASL_KEY_REF_PID**
Reference PID for messages proxied by launchd

asl.**ASL_KEY_REF_PROC**
Reference process for messages proxied by launchd

asl.**ASL_KEY_AUX_TITLE**
Auxiliary title string

asl.**ASL_KEY_AUX_UTI**
Auxiliary Uniform Type ID

asl.**ASL_KEY_AUX_URL**
Auxiliary Uniform Resource Locator

asl.**ASL_KEY_AUX_DATA**
Auxiliary in-line data

asl.**ASL_KEY_OPTION**
Internal

asl.**ASL_KEY_SENDER_INSTANCE**
Sender instance UUID.

asl.**ASL_KEY_SENDER_MACH_UUID**
Sender Mach-O UUID

asl.**ASL_KEY_FINAL_NOTIFICATION**
Syslogd posts value as a notification when message has been processed.

asl.**ASL_KEY_OS_ACTIVITY_ID**
Current OS Activity for the logging thread.

1.4.6 Match directions

asl.**ASL_MATCH_DIRECTION_FORWARD**
Match in forward direction.

asl.**ASL_MATCH_DIRECTION_REVERSE**
Match in reverse direction.

1.4.7 Message types

asl.**ASL_TYPE_UNDEF**
Undefined type

asl.**ASL_TYPE_MSG**
A regular log message.

asl.**ASL_TYPE_QUERY**
A query message.

asl.**ASL_TYPE_LIST**
A list of messages or queries

asl.**ASL_TYPE_FILE**
Abstraction for a ASL data file.

asl.**ASL_TYPE_STORE**
Abstraction for an ASL data store (directory containing data files)

asl.**ASL_TYPE_CLIENT**
High-level objet that abstracts ASL interactions.

1.4.8 Message encoding

asl.**ASL_ENCODE_NONE**
Don't escape characters

asl.**ASL_ENCODE_SAFE**
Escapes backspace as "^H", replaces carriage returns by newlines, add tabs after newlines to indent continued message text.

asl.**ASL_ENCODE_ASL**
Use C style encoding for a subset of non-printable characters, like the vis(1) command with the "-c" option.

asl.**ASL_ENCODE_XML**
Encode message as XML text.

1.4.9 File message formats

asl.**ASL_MSG_FMT_RAW**
Complete message structure with key-value pairs in square brackets.

asl.**ASL_MSG_FMT_STD**
Simular to `ASL_MSG_FMT_BSD`, but with message priority.

asl.**ASL_MSG_FMT_BSD**
Format used by the syslog daemon for system log files.

asl.**ASL_MSG_FMT_XML**
Formatted as XML property lists.

asl.**ASL_MSG_FMT_MSG**
(Undocumented format)

1.4.10 File time formats

asl.**ASL_TIME_FMT_SEC**
Show timestamps as seconds since the epoch.

asl.**ASL_TIME_FMT_UTC**
Show timestamps in UTC in the format "yyyy-mm-dd hh:mm:ssZ".

asl.**ASL_TIME_FMT_LCL**
Show timestaps in the local timezone with format "mmm dd hh:mm:ss".

1.4.11 Filter masks

These are used for client-side filtering.

asl.**ASL_FILTER_MASK_EMERG**

asl.**ASL_FILTER_MASK_ALERT**

asl.**ASL_FILTER_MASK_CRIT**

asl.**ASL_FILTER_MASK_ERR**

asl.**ASL_FILTER_MASK_WARNING**

asl.**ASL_FILTER_MASK_NOTICE**

asl.**ASL_FILTER_MASK_INFO**

asl.**ASL_FILTER_MASK_DEBUG**

1.4.12 Open options

asl.**ASL_OPT_STDERR**
Write a copy of log lines to the stderr stream.

asl.**ASL_OPT_NO_DELAY**
Immediately create a connection to the logging subsystem, instead of waiting for the first log message.

asl.**ASL_OPT_NO_REMOTE**
Ignore the server side log filter for messages send using this connection. Using this option requires root privileges.

1.4.13 File descriptor types

asl.**ASL_LOG_DESCRIPTOR_READ**
File descriptor is readable, ASL will read log lines from it.

asl.**ASL_LOG_DESCRIPTOR_WRITE**
File descriptor is writable. ASL will convert the file descriptor to another writable descriptor where the application can write lines that will be converted to log messages.

CHAPTER 2

Examples

2.1 Basic logging

```
cli = asl.aslclient("example", "user", 0)
cli.log(None, asl.ASL_LEVEL_NOTICE, "hello world")
```

2.2 Logging with more control

```
cli = asl.aslclient("example", "user", 0)
msg = asl.aslmsg(asl.ASL_TYPE_MSG)
msg[asl.ASL_KEY_FACILITY, "com.secrets.r.us");
msg[asl.ASL_KEY_LEVEL] = asl.ASL_STRING_EMERG
msg[asl.ASL_KEY_MSG] = "Breaking attempt!"

cli.send(msg)
```

2.3 Sending log messages to the default view of Console.app

```
cli = asl.aslclient("example", "com.apple.console", 0)
msg = asl.aslmsg(asl.ASL_TYPE_MSG)
msg[asl.ASL_KEY_FACILITY] = "com.apple.console"
msg[asl.ASL_KEY_LEVEL] = asl.ASL_STRING_NOTICE
msg[asl.ASL_KEY_READ_UID] = str(os.getuid())

cli.log(msg, asl.ASL_LEVEL_NOTICE, "hello console.app!")
```

2.4 Logging an auxiliary URL

```
cli = asl.aslclient("example", "com.apple.console", 0)
msg = asl.aslmsg(asl.ASL_TYPE_MSG)
msg[asl.ASL_KEY_FACILITY] = "com.apple.console"
msg[asl.ASL_KEY_LEVEL] = asl.ASL_STRING_NOTICE
msg[asl.ASL_KEY_READ_UID] = "-1"
msg[asl.ASL_KEY_MSG] = "Python was here"

asl.log_auxiliary_location(msg, "More information", None, "http://www.python.org/")
```

2.5 Querying the ASL database

```
cli = asl.aslclient("example", "user", 0)
msg = asl.aslmsg(asl.ASL_TYPE_QUERY)
msg.set_query(asl.ASL_KEY_FACILITY, "com.apple.console", asl.ASL_QUERY_OP_EQUAL)

for info in cli.search(msg):
    print info.asdict()
```

2.6 Integration with the logging package

This example implements a `logging.Handler` subclass that forwards all messages to ASL while marking them for display in Console.app's default view.

This class is not part of the API of the ASL package because it is at this time not clear if the implementation is fully usable in its current form.

```
import logging
import asl
import sys
import os

# Translation from logging levels to ASL levels:

_LOGGING2ASL = {
    logging.DEBUG: asl.ASL_STRING_DEBUG,
    logging.INFO: asl.ASL_STRING_INFO,
    logging.WARNING: asl.ASL_STRING_WARNING,
    logging.ERROR: asl.ASL_STRING_ERR,
    logging.CRITICAL: asl.ASL_STRING_CRIT,
    logging.FATAL: asl.ASL_STRING_ALERT,
}
def _logging2asl(lvl):
    try:
        return _LOGGING2ASL[lvl]
    except KeyError:
        r = asl.ASL_STRING_DEBUG
        for k in sorted(_LOGGING2ASL):
            if k < lvl:
                r = _LOGGING2ASL[k]
        return r
```

(continues on next page)

(continued from previous page)

```
#
# Define a logging handler:
#

class ASLConsoleHandler (logging.Handler):
    def __init__(self, ident=None, level=asl.ASL_STRING_INFO):
        logging.Handler.__init__(self)
        self._asl = asl.aslclient(ident, level, 0)

    def emit(self, record):
        msg = asl.aslmsg(asl.ASL_TYPE_MSG)

        # Add all attributes of the logging record
        # to the ASL log message:
        for k in dir(record):
            if k in ('args', 'levelno', 'levelname', 'msecs', 'relativeCreated',
↪'asctime', 'created'):
                continue
            if k.startswith('_'):
                continue

            # What about exc_info?

            msg["py." + k] = str(getattr(record, k))

        # Then set up the default attributes:
        msg[asl.ASL_KEY_FACILITY] = "com.apple.console"
        msg[asl.ASL_KEY_LEVEL] = _logging2asl(record.levelno)
        msg[asl.ASL_KEY_READ_UID] = str(os.getuid())
        msg[asl.ASL_KEY_MSG] = self.format(record)

        self._asl.send(msg)


# Use the logger class:

logging.basicConfig(stream=sys.stderr, level=logging.DEBUG)
root = logging.getLogger()
print root
root.addHandler(ASLConsoleHandler())

root.warning("test me")
```

CHAPTER 3

Module as a script

The module can be used as a script:

```
$ python3 -m asl --help
```

This is primarily intended as an example on how to use the module, but can be usefull on its own.

3.1 Usage

3.1.1 Writing messages to the Console.app default view

```
$ python3 -m asl consolelog [-i IDENT] [-l LEVEL] message ...
```

This logs the message arguments as a single line (separated by spaces) with the attributes needed to end up in the default view of Console.app.

The *IDENT* is the source identifier and is usually an application name. It defaults to the name of the user.

The *LEVEL* is the log level, and is one of “Emergency”, “Alert”, “Critical”, “Error”, “Warning”, “Notice”, “Info” and “Debug” (from high to low priority). The default is “Notice”, which is the lowest priority that will end up in the default view of Console.app.

3.1.2 Generic logging

```
$ python3 -m asl writelog {-k KEY VALUE}...
```

Write a log message to the ASL, with all message parameters under control of the user.

The *IDENT* is the source identifier and is usually an application name. It defaults to the name of the user.

The *FACILITY* is the log facility.

The *LEVEL* is the log level, and is one of "Emergency", "Alert", "Critical", "Error", "Warning", "Notice", "Info" and "Debug" (from high to low priority). The default is "Notice".

The "-k" option is used to set the attributes of the log message. You should at least include a "Message" key with the text of the log message.

The standard keys are:

- *Time*: Time of logging (added by the ASL daemon)
- *TimeNanoSec* Nano-second resolution time (added by the ASL daemon)
- *Host*: Host where the message was written
- *Sender*: Sender of the log message
- *Facility*: Log facility

  This is either one of the regular syslog facilities (auth, authpriv, cron, ..., or an arbitrary string. Messages that should be shown in the default view of Console.app should use facility "com.apple.console".
- *PID*: Process ID of the sending proces (replaced by the ASL daemon)
- *UID*: UID of the sending process (replaced by the ASL daemon)
- *GID*: GUID of the sending proccess (replaced by the ASL daemon)
- *Level*: Log level (see the -l command of the *consolelog* command)
- *Message*: Text of the log message
- *ReadUID*: UID that may read the message, use "-1" for "all users"
- *ReadGID*: GID that may read the message, use "-1" for "all groups"
- *ASLExpireTime*: Expiry time for this message (default to 7 days after the log time)
- *ASLMessageID* Message ID for this message (replaced by the ASL daemon)
- *Session*: Session ID
- *RefPID* Reference PID for messages proxied by launchd
- *RefProc*: Reference process for messages proxied by launchd
- *ASLAuxTitle*: Auxiliary title string
- *ASLAuxUTI*: Auxiliary UTI
- *ASLAuxURL*: Auxiliary URL
- *ASLAuxData*: Auxiliary in-line data
- *SenderInstance*: Sender instance UUID

3.1.3 Query the ASL database

```
python3 -m asl query [-f FMT] [--format FMT] [-C] {-e KEY}... {-k KEY OP VALUE}...
```

Search the ASL database for matching records. All found records are printed on stdout. By default all records are printed, and using the "-C", "-e" and "-k" options a subset can be selected:

When mutiple "-C", "-e" and "-k" are present all of them must match (that is, the subexpressions are combined with an "AND" operator).

- "-f FMT" or "–format FMT"

  Use FMT for formatting the records before printing them. The format string is a format string that can be using with `str.format`, and it will be used in such a way that non-existing keys are ignored.

  When this option is not used all attributes of records are printed and records are separated by a single empty line.

- "-C"

  Add selection for messages destined for the Console.app. This is equivalent to "-k Facility eq com.apple.console"

- "-e KEY"

  Add a test that checks that attribute *KEY* exists.

- "-k KEY OP VALUE"

  Add a test for the value of attribute *KEY*.

  The following operators are recognized:

  - *eq*: The value must be equal to the specified value
  - *ne*: The value must not be equal to the specified value
  - *gt*: The value must be lexographically greater than the specified value
  - *ge*: The value must be lexographically greater than or equal to the specified value
  - *lt*: The value must be lexographically less than the specified value
  - *le*: The value must be lexographically less than or equal to the specified value
  - *match*: The value must match the specified regular expression (see regex(3))
  - *startswith*: The value must start with the specified value
  - *endswith*: The value must end with the specified value
  - *contains*: The value must contain the specified value
  - ==: The value must be a number and must be equal to the specified value
  - *!*=: The value must be a number and must be different from the specified value
  - >: The value must be a number and must be greater than the specified value
  - >=: The value must be a number and must be greater than or equal to the specified value
  - <: The value must be a number and must be less than the specified value
  - <=: The value must be a number and must be less than or equal to the specified value

  The string operators can be prefix with "C" to perform case folding before the comparison.

  See the section on *Generic logging* for a description of the standard keys.

CHAPTER 4

Release history

4.1 asl 1.1

- Removed support for Python 2.7 and older Python 3 versions
- Added support for Python 3.6, 3.7 and 3.8
- Migrated to GitHub
- Added type annotations
- Better code quality through flake8, isort, black and mypy

4.2 asl 1.0.2

- Fixes some packaging issues
- Updates constants for macOS 10.12

4.3 asl 1.0

- First public release
- Adds command-line interface (`python3 -m asl --help`)
- Add translation from numeric ID to string, and the reverse, for log levels
- Updated the documentation

4.4 asl 0.9

- Initial release with support for the ASL API in OSX 10.8.

CHAPTER 5

Online Resources

- Sourcecode repository on GitHub
- The issue tracker

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

Symbols

A

C

K

L

O

R

S