Revision date: March 1, 2008

Grid Logging: Best Practices Guide

Brian Tierney, Dan Gunter: Lawrence Berkeley National Lab

Laura Pearlman: ISI

current version can be found at:
http://www.cedps.net/index.php/LoggingBestPractices

Introduction

The purpose of this document is to help developers of Grid middleware and application software generate log files that will be useful to Grid administrators, users, developers and Grid middleware itself. Currently, most of the currently generated log files are only useful to the author of the program. Good logging practices are instrumental to performance analysis, problem diagnosis, and security auditing tasks such as incident tracing and damage assessment.

This document does not discuss the issue of a logging API. It is assumed that a standard log API such as syslog (C), log4j (Java), or logger (Python) is being used. Other custom logging API or even *printf* could be used. The key point is that the logs must contain the required information in the required format.

At a high level of abstraction, the best practices for Grid logging are:

- Consistently structured, typed, log events
- A standard high-resolution timestamp
- Use of logging levels and categories to separate logs by detail and purpose.
- Consistent use of global and local identifiers.
- Use of some regular, newline-delimited ASCII text format

The rest of this document describes each of these recommendations in detail.

Log event model

Discussion

Common practice is to model logged events as (natural language) sentences. We wish to define an alternate model that can be used for the subset of log messages most useful for automated analysis – entry and exit from regions, success and failure of important operations, etc. – and that will be far easier to deal with programmatically.

This section studiously avoids mentioning the actual concrete representation of logged events following this model. Later sections will define some general log conventions, and also provide a full implementation of the model.

Base model

Each log event consists of:

- an event name, which defines the event type
- a set of attributes and values
    - the set of attributes are determined by the type
    - the values are not explicitly typed

Database aficionados might notice that this definition corresponds closely with the relational model's definition of an *n-tuple* and its concrete cousin, a row in a database table: the "type" corresponds to the table name, the attribute names to the column names, and the values to (of course) the values.

Extension for start and end events

Log events are associated with program activities, such as "reading data from disk" or "authorizing a user". Our model distinguishes between log events that encompass all of an activity, and those that are at the *start* or at the *end* of the activity. If they are at the start or end, logged events should explicitly record this in the event name. For example, "disk.read.start" and "disk.read.end". By not indicating whether a log event is at the start or the end, the implication is that the activity already occurred, but that the *start* of the activity could not be (easily) logged, and that therefore this event is the only one available. Thus, we extend the base model:

- *base:* an event name, which defines the event type
- *extended:* an event name, which defines the event type and whether the event is at the start or end of the associated activity, or after an activity with an unknown start

We recommend that if an activity is going to be logged at all, where possible both the start and the end of the activity should be logged; failures being just another kind of "end" of an activity. This makes it particularly easy to isolate stalls and failures in a given workflow.

Log event representation

Discussion

This document does not proscribe a particular log format. However, it does assume that log messages must meet some basic practical requirements:

- human "readable" in a terminal or text editor
- directly compatible with *syslog* and *grep*

These requirements, in our view, are fundamental and rule out "binary" formats. More precisely, log event representations should:

- use 7-bit ASCII characters only
- use newlines (ASCII 0x0A, "\n") as the log event delimiter
- use English names and decimal numbers

There are some common elements whose representation is also worth standardizing at this point: event type names, timestamps, and the reserved attribute names.

Event names

The name of the event defines an event type. To avoid name clashes when logs from independent programs are combined, the event name should be in a namespace that is unique to the generating program. This namespace should be hierarchical, from general to specific, and we recommend its prefix be based on DNS. For example, for a read from disk in the program "foo" belonging to "my.company.org", good start and end event names might be: `org.company.my.foo.disk.read.start` *and* `org.company.my.foo.disk.read.end`.

Examples

Some example event names include:

- org.globus.gridFTP.start
- org.globus.gridFTP.authn.x509.start
- org.globus.gridFTP.authn.x509.end
- org.globus.gridFTP.transfer.start
- org.globus.gridFTP.transfer.end
- org.globus.gridFTP.end

Attributes

Only two attributes are required: the name of the event, and a timestamp.

Timestamp representation

We recommend a timestamp representation that is a highly readable variant of the ISO8601 time standard [1]:

*YYYY-MM-DD*T*HH*:*MM*:*SS*.*SSSSSS*Z

For example: 2000-10-26T08:34:26.30323Z. The “Z” at the end signifies “UTC”, and is highly recommended, although +/- offset from GMT is also allowed in place of the “Z”.

Microsecond timestamp resolution, where possible, is highly recommended. It's easy to ignore extra digits but very hard to add them later.

Other Attributes

Attribute names, in general, do not need to be hierarchical like event names; they implicitly within a namespace defined by that event name. We have found it useful to define some common attributes that are the same across all log events:

- level – logging level

- status – integer status code, for example 0 for success and -1 for failure
- guid – global unique identifier, see §Log event identifiers, below
- prog – program name
- DN – X.509 distinguished name
- msg – error/status message string

Examples

While we do not proscribe a particular log format, we have found that using a set of self-describing values formatted as name-value pairs is useful and easy to parse. For this remainder of this document we use this format for sample logs.

For example:

```
ts=2006-12-08T18:39:19.372375Z org.my.TBS.job.submit.start jobId=37900
ts=2006-12-08T18:39:23.114369Z org.my.TBS.job.submit.end jobId=37900
      status=0
```

The addition of log file grammar such as the name-value pair structure encourages more regular and normalized representations than natural language sentences commonly found in ad-hoc logs. For example the equivalent of:

```
error: read from socket on foobar.org:1234, remote host baz.org:4321
returned -1
```

could instead be:

```
ts=2006-12-08T18:48:27.598448Z event=org.my.myapp.socket.read.end
level=ERROR status=-1 host=foobar.org:1234 peer=baz.org:4321
```

For the name part of each name-value pair, we suggest adopting the Java method naming convention of a lowercase first word, and capitalized letter on each successive word, also known as lowerCamelCase.

Log event identifiers

Discussion

It is crucial in a Grid logging environment to be able to associate distributed events. Therefore all logging events related a particular program or thread instantiation must contain global identifier that relates it to other log events in the enclosing scope (even if this is the only event in that scope). Unlike event names, these identifiers indicate which particular instance of the containing task is logging the event, and effort should be made to guarantee that they are indeed unique to that instance.

We recommend using the standard "globally unique identifier" (guid) for this. Rules for constructing a guid are given by RFC4122[3], and implemented by several freely available libraries and the uuidgen program (a standard utility on Windows and Unix). Guid's should be encoded in the string representation given by RFC4122, e.g. "5006F942-B782-4D86-9E66-63EB03269FD0". For situations where it is important to

have a smaller identifier, one can optionally use a local identifier such as a request or transaction number.

Ideally all components from the entire workflow would use the same guid, but since is may be difficult to pass this ID between the various components, just using a unique ID with a given component on a give host is allowed.

Due to the verbosity of this approach, it is reasonable to not include such identifiers in every message at DEBUG and TRACE logging levels. However, thought should be given as to how these events can be associated with other events that do have unique identifiers, so they can be added post-hoc if needed.

Logging guidance

Service initiation, configuration and termination

Whenever a service starts up or a service request thread is launched, this should be logged. If the service can be configured, this message must contain a reference to the service configuration used. The termination message should include a status or termination message or code.

Errors

All errors that cause a component to exit must be logged.

Remote Connections

When a log message pertains to an attempted connection to/from a remote service, the log should contain the IP address and port number. This is particularly important for tracking down firewall issues.

Authentication and Authorization Logging

One of the most common types of errors related to Grid computing have to do with authentication and authorization problems. Therefore these are some of the most important log events, and should contain as much information as possible. As with all other operations, both authn.start and authn.end should be logged, with errors included in the authn.end event.

Authorization log entries should include the standard log info plus:

- the authentication method
- the claimed identity (if available)
- a reason code that describes the reason for the authentication error. It may make sense to define different reason codes for different authentication methods – for example, standard X.509 authentication may have reasons like “certificate expired” or “certificate issued by untrusted CA”, while username/password authentication may have reasons like “unknown user” or “bad password”.

- An optional text string with more information.

For example:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.authn.x509.start
   DN="/O=CEDS/CN=Some User" guid=E8500036-4EBE-4D39-95BB-
   0AC8DDE66903
```

Success:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.authn.x509.end
   DN="/O=CEDS/CN=Some User" guid=E8500036-4EBE-4D39-95BB-
   0AC8DDE66903 status=0
```

Fail:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.authn.x509.end
   status=-1  DN="/O=CEDS/CN=Some User" reason="untrusted CA"
   msg="Certificate Authority '/O=CEDS/CN=Certificate Authority' is
   not a trusted certificate authority" guid=E8500036-4EBE-4D39-
   95BB-0AC8DDE66903 level=ERROR
```

Authentication errors – that is, errors in the authentication system itself-- should always be logged. These log entries should contain essentially the same attributes as authentication failures, but they should probably be logged at a higher priority than authentication failures because they may indicate a problem that needs immediate attention. For example:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.authn.x509.end
   level=CRITICAL DN="/O=CEDS/CN=Some User" msg="Cannot open CA
   certificate file" file=/etc/grid-security/certificates/4a6cd8b1.0
   guid=E8500036-4EBE-4D39-95BB-0AC8DDE66903 status=-1
```

Authorization Events:

Authorization policies sometimes involve a combination of authorization methods; for example, "allow anyone to do operation X if they are in the gridmap file or if they are in group G as authorized by a VOMS server, but not if they're not listed in the local blacklist". Each of these authorization steps should be logged, along with the following information:

- the remote identity
- the authorization mechanism (gridmap, "none", contacting a remote authorization server, etc.)
- any mechanism-specific attributes

For example:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.authz.gridmap.start
   DN="/O=CEDS/CN=Some User" guid=E8500036-4EBE-4D39-95BB-0AC8DDE66903
ts=2006-12-08T18:39:23.114369Z event=org.globus.authz.gridmap.end
   DN="/O=CEDS/CN=Some User" status=100 guid=E8500036-4EBE-4D39-95BB-
   0AC8DDE66903

ts=2006-12-08T18:39:23.114369Z event=org.globus.authz.VOMS.start
   DN="/O=CEDS/CN=Some User" operation=runJob guid=E8500036-4EBE-4D39-
   95BB-0AC8DDE66903
```

```
ts=2006-12-08T18:39:23.114369Z gatekeeper.authz.VOMS.end
   DN="/O=CEDS/CN=Some User" status=0 guid=E8500036-4EBE-4D39-95BB-
   0AC8DDE66903

ts=2006-12-08T18:39:23.114369Z
   event=org.globus.authz.localBlackList.start guid=E8500036-4EBE-4D39-
   95BB-0AC8DDE66903 DN="/O=CEDS/CN=Some User"
ts=2006-12-08T18:39:23.114369Z
   event=org.globus.authz.localBlackList.end guid=E8500036-4EBE-4D39-
   95BB-0AC8DDE66903 DN="/O=CEDS/CN=Some User" status=0
```

Authorization Errors

Authorization errors should include the following:

- the remote identity
- the authorization mechanism (gridmap, contacting a remote authorization server, etc.)
- any mechanism-specific attributes

For example:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.authz.VOMS.end status=-
   1 level=CRITICAL DN="/O=CEDS/CN=Some User" msg="VOMS service
   unreachable" guid=F7D64975-069A-4152-A21F-57109AA46DFA

ts=2006-12-08T18:39:23.114369Z event=org.globus.authz.gridmap.error
   status=-1 level=CRITICAL DN="/O=CEDS/CN=Some User" msg="Cannot open
   gridmap file for reading" file=/etc/grid-security/grid-mapfile
   guid=F7D64975-069A-4152-A21F-57109AA46DFA
```

Acknowledgements

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contract number DE-AC02-05CH11231.

References

1. ISO-8601, "Data Elements and Interchange Formats - Information Exchange - Representation of Dates and Times", International Organization for Standardization, 1888 http://www.iso.ch/markete/8601.pdf
2. Dan Gunter, James Magowan. "An analysis of "Top N" Event Descriptions", GGF Informational document: GFD-I.025
3. P. Leach, M. Mealling, and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC-4122, July 2005, http://www.ietf.org/rfc/rfc4122.txt
4. C. Lonvick, "The BSD Syslog Protocol", IETF RFC3164, http://www.ietf.org/rfc/rfc3164.txt

Appendix A. Log event representation (full)

Discussion

In our own work, and examples throughout this text, we use a name=value pair format for log events. For example:

```
ts=2006-12-08T18:39:19.372375Z event=org.my.sample.event value=0
```

As of this writing, this format is integrated into the Globus Project codebase and will be on by default in the GT 4.2 release.

Grammar

```
log         = <nvp>*
line        = <ts> & <event> & ( <nvp> )*
nvp         = <name> "=" <value>
ts          = // see Timestamp section
event       = "event=" <name>
name        = ( {alphanums + "-_." } )*
value       = ( <not-space> )* | <quote> <not-newline>* <quote>
quote       = "
space       = " "
not-space   = {printable 7-bit ASCII characters except <space> and
   "\n"}
not-newline = {printable 7-bit ASCII characters except "\n"}
```

The important point of this section is that all grid logs should have *some* grammar, preferably one that is simple and regular, and easily (or even automatically) translated to parsing code.

Appendix B. Log example for Grid Workflow

This section contains some selected log entries for a grid workflow where a client stages data, submits a job, then pulls back the results. The sample log entries leave out most of the details about which URL is being transferred or which job is being run; instead it focuses on the identifiers that would need to be passed around.

RFT client request:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.rft.client.put.start
   guid=27DC13A3-202E-426B-BBEE-06DAA482340B
```

RFT server:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.rft.server.put.start
   guid=27DC13A3-202E-426B-BBEE-06DAA482340B
Authorize:
ts=2006-12-08T18:39:23.114369Z event=org.globus.rft.server.authn.start
   guid=27DC13A3-202E-426B-BBEE-06DAA482340B
ts=2006-12-08T18:39:23.114369Z event=org.globus.rft.server.authn.end
   guid=27DC13A3-202E-426B-BBEE-06DAA482340B status=0
```

```
Do the transfer (e.g. with GridFTP) :
ts=2006-12-08T18:39:23.114369Z
   event=org.globus.rft.server.transfer.start guid=27DC13A3-202E-426B-
   BBEE-06DAA482340B
ts=2006-12-08T18:39:23.114369Z
   event=org.globus.rft.server.transfer.start guid=27DC13A3-202E-426B-
   BBEE-06DAA482340B
ts=2006-12-08T18:39:23.114369Z event=org.globus.rft.server.put.end
   guid=27DC13A3-202E-426B-BBEE-06DAA482340B status=0
ts=2006-12-08T18:39:23.114369Z event=org.globus.rft.client.put.end
   status=0 guid=27DC13A3-202E-426B-BBEE-06DAA482340B status=0
```

Submit job through globus-job-run:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.globus-job-run.start
   guid=27DC13A3-202E-426B-BBEE-06DAA482340B
```

Gatekeeper `authorization:`

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.gatekeeper.authn.start
   guid=27DC13A3-202E-426B-BBEE-06DAA482340B
ts=2006-12-08T18:39:23.114369Z event=org.globus.gatekeeper.authn.end
   guid=27DC13A3-202E-426B-BBEE-06DAA482340B status=0
```

Gatekeeper `calls JobManager:`

```
ts=2006-12-08T18:39:23.114369Z
   event=org.globus.jobmanager.request.start guid=27DC13A3-202E-426B-
   BBEE-06DAA482340B
```

JobManager runs job:

```
ts=2006-12-08T18:39:23.114369Z
   event=org.globus.jobmanager.response.start guid=27DC13A3-202E-426B-
   BBEE-06DAA482340B
ts=2006-12-08T18:39:23.114369Z
   event=org.globus.jobmanager.program.start guid=27DC13A3-202E-426B-
   BBEE-06DAA482340B
ts=2006-12-08T18:39:23.114369Z event=org.globus.jobmanager.program.end
   status=0 guid=27DC13A3-202E-426B-BBEE-06DAA482340B
```

End of Gatekeeper request to JobManager:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.jobmanager.request.end
   status=0 guid=27DC13A3-202E-426B-BBEE-06DAA482340B
```

End of job run:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.globus-job-run.end
   status=0 guid=27DC13A3-202E-426B-BBEE-06DAA482340B
```

Sample Log Events

Globus Gatekeeper:

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.gatekeeper.start
    guid=55A0DF43-D7AB-40BF-A023-04444B93F76E
    remoteHost=gridhost.yoursite.gov:NNNN localHost=myhost.foo.gov:NNNN
    requestType=jobSubmit
ts=2006-12-08T18:39:23.114369Z
    event=org.globus.gatekeeper.authn.x509.start guid=55A0DF43-D7AB-
    40BF-A023-04444B93F76E DN="/O=CEDS/CN=Some User"
ts=2006-12-08T18:39:23.114369Z
    event=org.globus.gatekeeper.authn.x509.end guid=55A0DF43-D7AB-40BF-
    A023-04444B93F76E DN="/O=CEDS/CN=Some User" status=0
ts=2006-12-08T18:39:23.114369Z
   event=org.globus.gatekeeper.authz.GUMS.start guid=55A0DF43-D7AB-
   40BF-A023-04444B93F76E DN="/O=CEDS/CN=Some User"
   mappingService="https://cmssrv08.fnal.gov:8443/gums/services/
   GUMSAuthorizationServicePort"
ts=2006-12-08T18:39:23.114369Z
    event=org.globus.gatekeeper.authz.GUMS.end guid=55A0DF43-D7AB-40BF-
    A023-04444B93F76E DN="/O=CEDS/CN=Some User" localUser=gridex
    localUID=10657 localGID=10657 gridSecurityHTTPBodyFD=8 status=0
ts=2006-12-08T18:39:23.114369Z
    event=org.globus.gatekeeper.jobManager.start guid=55A0DF43-D7AB-
    40BF-A023-04444B93F76E path="/opt/globus/libexec/globus-job-manager"
    gatekeeperJmId="2006-11-09.00:06:28.0000006577.0000000000"
    executionHost=128.105.121.51 securityContextFD=11 childID=6639
ts=2006-12-08T18:39:23.114369Z
    event=org.globus.gatekeeper.jobManager.end guid=55A0DF43-D7AB-40BF-
    A023-04444B93F76E status=0
ts=2006-12-08T18:39:23.114369Z event=org.globus.gatekeeper.end
    guid=55A0DF43-D7AB-40BF-A023-04444B93F76E status=0
```

Globus GridFTP Server

```
ts=2006-12-08T18:39:23.114369Z event=org.globus.gridFTP.start
   prog=GridFTP-4.0.3 localHost=myhost remoteHost=somehost.gov:56010
   serverMode=inetd guid=D4E922E1-6F7D-4752-B126-013BFE71C80B
ts=2006-12-08T18:39:23.114567Z
   event=org.globus.gridFTP.authn.x509.start
   DN="/DC=org/DC=doegrids/OU=People/CN=Somebody" guid=D4E922E1-6F7D-
   4752-B126-013BFE71C80B
ts=2006-12-08T18:39:23.114567Z event=org.globus.gridFTP.authn.x509.end
   DN="/DC=org/DC=doegrids/OU=People/CN=Somebody" guid=D4E922E1-6F7D-
   4752-B126-013BFE71C80B status=0
ts=2006-12-08T18:39:23.114567Z
   event=org.globus.gridFTP.authz.gridmap.start
   DN="/DC=org/DC=doegrids/OU=People/CN=Somebody" guid=D4E922E1-6F7D-
   4752-B126-013BFE71C80B
ts=2006-12-08T18:39:25.514369Z
   event=org.globus.gridFTP.authz.gridmap.end
   DN="/DC=org/DC=doegrids/OU=People/CN=Somebody"
   localUser=uscmspool381 guid=D4E922E1-6F7D-4752-B126-013BFE71C80B
   status=0
```

```
ts=2006-12-08T18:39:25.864369Z event=org.globus.gridFTP.transfer.start
   infile=/tmp/myfile tcpBufferSize=128KB dataBlockSize=262144
   numStreams=1 numStripes=1 destHost=129.79.4.64 guid=D4E922E1-6F7D-
   4752-B126-013BFE71C80B
ts=2006-12-08T18:45:02.214369Z event=org.globus.gridFTP.transfer.end
   infile=/tmp/myfile bytesTransferred=678433 guid=D4E922E1-6F7D-4752-
   B126-013BFE71C80B status=0
ts=2006-12-08T18:45:02.214386Z event=org.globus.gridFTP.end
   guid=D4E922E1-6F7D-4752-B126-013BFE71C80B status=226
```