Logging in JRA1 Code

EGEE07, Budapest, 1-5 Oct, 2007

Steve Fisher/RAL and Ales Krenek/Cesnet

www.eu-egee.org
www.glite.org

Information Society and Media

Overview

- **Brief history of:**
  "Recommendations (and a few rules) for logging in JRA1 Code"
  - https://twiki.cern.ch/twiki/pub/EGEE/EGEEgLite/logging.html
- **Aims of the document**
- **Specific recommendations/rules**
- **Conclusion/summary**

History

- **Have discussed several times within JRA1:**
  - Each time the decision was to adopt a logging framework
- **Re-triggered by David Groep's document: Middleware Security Audit Logging Guidelines (version 0.8)**
  - This document says what should be logged (anything security related) and how it should be logged (via syslog)
- **TCG Top ROC Issue#12**
  - "Provide better and clearer log system, with a standard logging format…"
- **We (Ales K and Steve F) were asked to produce a recommendation**
  - Initial document discussed primarily within JRA1
  - Current document (version 1.1)
    - Less strict than the earlier drafts
    - However it has been accepted by JRA1.
    - Note the yellow bits

Logging – for whom?

- **Logging serves multiple purposes:**
  - developers
    - to check program logic
    - to understand problems on a running system
  - sys-admins
    - to check for correct functioning and diagnose problems
    - to investigate possible security incidents

Aims

- **Unify configuration of logging**
  - This is valuable for smooth deployment of the services
  - Other requirements follow from it
- **Follow the spirit of the security audit logging guidelines document**
- **Uniformity of what is logged**
  - makes life easier for site administrators
  - improves communication between developers when interaction between middleware components is analysed.
- **Minimise number of solutions**
  - With diverse developer communities we are unlikely to find a solution which suits everybody

Common Configuration Mechanism

- **Sys-admins determine what is logged at their site**
- **If sys-admin suspects a problem he may want to increase the logging**
- **Should be easy to do this without intimate knowledge of the middleware**
- **Cannot yet insist that all JRA1 components should use a logging mechanism with a configuration file based on log4j.**
  - Practical problems with available libraries
  - Situation is getting better!

Specific libraries to consider

- **Java**
  - either log4j or Jakarta commons
- **C++**
  - log4cxx
    - Apache release 0.10.0 is imminent
  - log4cpp
    - Currently no syslog appender
- **C**
  - log4c
    - Uses GCC extensions (v 1.2.0)
- **Python**
  - logger
    - Part of standard distribution – slightly non-standard config file
  - log4py
    - Precedes logger

The logger

- Logger has a name and is programmer view of logging
- Relation to appender (e.g. syslog or console appender) defined in configuration
- Logger has operations such as trace(), warn(), debug() to log message with specified severity

Set Of Loggers

A dedicated *security* logger MUST be used for security related messages

A dedicated *access* logger MUST be used to log initial access to a service

A dedicated *control* logger MUST be used for service startup, configuration and shutdown

Other loggers SHOULD be named in shallow hierarchy.

- **If you use tomcat this handles the access log**

Severity

- ***FATAL***
  - *The server/service/component is shutting down*
- ***ERROR***
  - *An unanticipated failure (i.e. a bug), or where the system is unable to behave according to its specification*
- ***WARN***
  - *Anticipated failure typically caused by bad user input. Note that from a system perspective bad user input is not an error.*
- ***INFO***
  - *Reporting of an operation executed by the server. The level of detail should be approximately the same as the specification and where appropriate use the same terminology. Any given operation should only be reported once at this level (after completion).*
- ***DEBUG***
  - *Reporting of an implementation-level operation. This may include details beyond what is described in the specification and an operation may generate more than one message at this level. Try to avoid having too much at the DEBUG level as it renders the code unreadable.*

Format of Message String

- **Brief but include as much relevant information as practical**
- **Only log at most 768 bytes to control, security and access loggers**
  - Some syslog implementations cannot deal with messages of more than 1024 bytes,
- **Any key-value pairs should be separated by a comma (which is preferred) or a space; and the key and value should be separated by an "="**
- **White space should be used as necessary for readability**
- **Messages must be self-contained.**
  - If your message is too long devise a way to split it into multiple stand-alone messages

What to log

Service start up, configuration and termination

To control logger

Authorization logging

To security logger

Initial service access, session establishment and termination

To access logger

Other Logging

As the developer sees fit

No logging in client APIs

It adds dependencies and

Makes porting of the API harder

Message Context

- If the request processing etc. is related to an externally identifiable entity (jobid, file guid, etc.) this identifier should be logged.
- Whenever there is a logical notion of session, request, etc., and multiple messages related to one such entity instance can be logged, it is recommended that a unique id is assigned to such an entity and it is logged in all the messages consistently.

Adding context information may have a big impact on the code and so should only be added when the code is undergoing major revision

Last slide

- SA1 should see steadily improved logging messages
- Messages can be routed via syslog if so desired
- Need JRA1 to identify preferred logging library for each service implementation language
- Following JRA1 agreement on the choice of libraries, we should be able to move to a common logging configuration mechanism at some stage.