



Enabling Grids for E-scienceE

# Logging and Reliability

*WLCG Service Reliability Workshop,  
CERN, Nov 2007*

*Steve Fisher/RAL*

[www.eu-egee.org](http://www.eu-egee.org)  
[www.glite.org](http://www.glite.org)



- **“Recommendations (and a few rules) for logging in JRA1 Code”**
  - <https://twiki.cern.ch/twiki/pub/EGEE/EGEEgLite/logging.html>
- **Specific recommendations/rules**
- **Reliability?**
- **Conclusion/summary**

- **TCG Top ROC Issue#12**
  - “Provide better and clearer log system, with a standard logging format...”
- **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

- **Unify configuration of logging**
  - This is valuable for smooth deployment of the services
- **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**
  - One library per programming language

- **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**
- **Short term target:**
  - One (or less) configuration file format per programming language

- **Java**
  - Log4j
- **C++**
  - Log4cpp
- **Python**
  - Logger
- **C**
  - ?

- **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**

- **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**



- **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.*

- **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
  
- **Brief but include as much relevant information as practical**

- **Logging does not directly provide reliability but:**
  - it provides a means to understand what the software is doing
  - permits giving sufficient information back to the developers to understand what has gone wrong and fix it

- **SA1 should see steadily improved logging messages**
- **Messages can be routed via syslog if so desired**
- **Logging contributes indirectly to reliability**