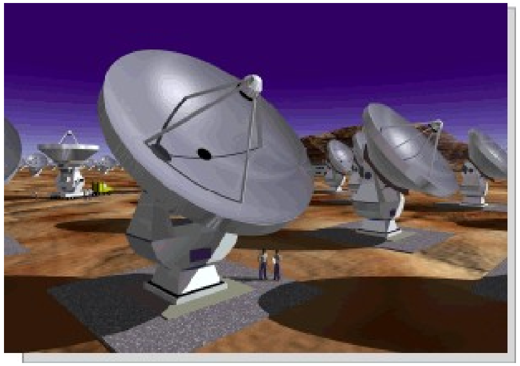


Atacama
Large
Millimeter
Array

ACS
Logging System Concepts and Example

J. Avarias & A. Caproni

(Restructured, based on slides from previous years)

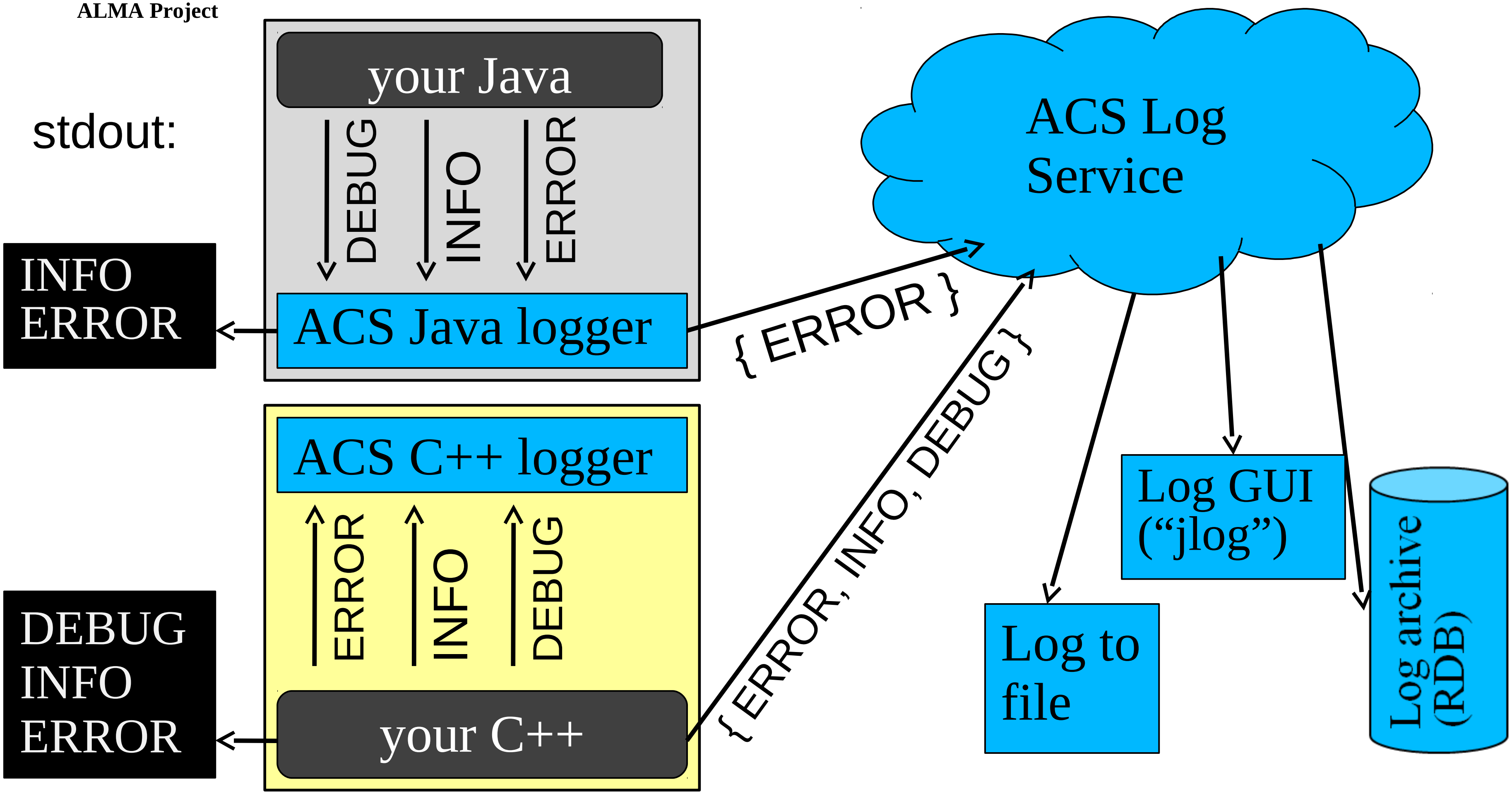
ALMA Project

Why Logging?

- ACS itself, and the software built with ACS, produce logs.
- The purpose of logs is to publish any kind of status and diagnostic information for interested clients and archival.
  - – Logs are essential for post-mortem analysis of hardware or software problems.
  - – For normal logs, the intended audience are developers.
  - – Optional “audience” set for individual logs. It can be used for filtering of logs, orthogonal to log levels.
    - ENGINEER
    - OPERATOR
    - SCILOG

ALMA Project

Conceptual Overview



stdout:

your Java

DEBUG

INFO

ERROR

ACS Java logger

INFO
ERROR

ACS C++ logger

ERROR

INFO

DEBUG

your C++

DEBUG
INFO
ERROR

{ ERROR }

{ ERROR, INFO, DEBUG }

ACS Log Service

Log to file

Log GUI ("jlog")

Log archive (RDB)

Log Forwarding

- Logs can be output locally on the console.
- Logs can be sent to a central Log service, coming from many components / clients in different languages (C++, Java, Py).
- From the Log service, all logs can be viewed together and analyzed live in a GUI, or written to file, or to a database and be processed later.

ALMA Project

Logger Objects & Filters

- Containers, components and some larger tools use separate ACS logger objects.
- Logs are prioritized with log levels, which allows filtering, to process only sufficiently important logs.
- Filtering can be configured per process or per logger.
- Log level filters are applied at the source, before the log is sent to the network.
- Log receiver tools have additional throw-away filtering capabilities, which work the same for all logs.

Log system technology (1)

- From the log producers to the Log service, and from there to the receiving clients, logs are transmitted in XML format.
- The ACS central Log service
  - Receives batches of log records through its IDL interface.
  - Is based on the CORBA (“Telecom”) Log service.
  - Forwards to a CORBA Notification Channel (NC), to deliver logs to registered clients.
  - That NC runs in a separate process, to not interfere with the applications’ notification channels.

Log Repeat Guards

- ACS offers “repeat guard” classes that can be used to conveniently reduce the number of executions of some identical activity such as logging.
  - Repeat guards are configurable based on the number of skipped executions, or skipping time, or combinations.
  - There are both log-specific and generic repeat guards.
- Real world example (from the jDAL, skipping logs for the same recordID). Here we use the advanced class MultipleRepeatGuard and wrap the logging of our message with it. This is different from using a RepeatGuardLogger, which does not construct its own log message.

```
MultipleRepeatGuard recordNotExistLogRepeatGuard = new …;
if (recordNotExistLogRepeatGuard.checkAndIncrement(recordID)) {
    String msg = “Record '" + recordID + "' does not exist.“;
    int repeatCount = recordNotExistLogRepeatGuard.
                                     counterAtLastExecution(recordID);
    if (repeatCount > 1) {
        msg += " (" + repeatCount + " identical logs reduced to this log)“;
    }
    m_logger.log(AcsLogLevel.NOTICE, msg);
}
```

ALMA Project

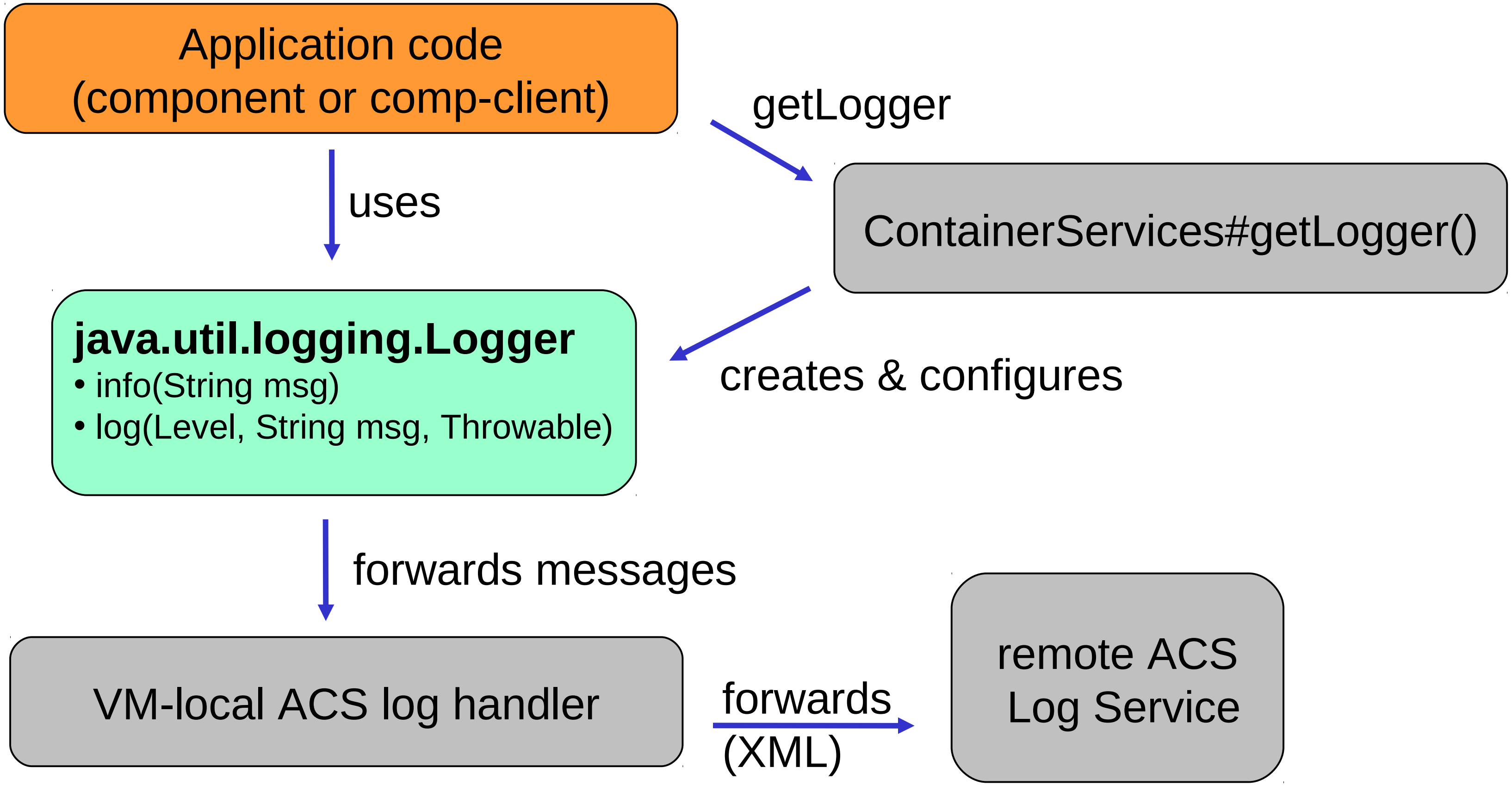
Log system technology (2)

- ACS logger objects appear like the “native” APIs
  - – C++ ACE logging
  - – Java: java.util.logging
  - – JDK-inspired Python logging API

ALMA Project

Log system technology (3)

Plain JDK Logger with ACS behind the scenes



Application code
(component or comp-client)

uses

getLogger

ContainerServices#getLogger()

creates & configures

java.util.logging.Logger
- info(String msg)
- log(Level, String msg, Throwable)

forwards messages

VM-local ACS log handler

forwards
(XML)

remote ACS
Log Service

ALMA Project

Log system technology (4)
XML syntax of a log entry

- Log entry type (*Debug*, *Info*, *Trace*...)
- Timestamp (ISO format)
- Source code info (*File*, *Line*, *Routine*)
- Runtime context info (*Host*, *Process*, *Thread*, *StackLevel*)
- Identification (*LogId*, *Uri*)
- Priority (1-11; default is 2)
- Data (<Name, Value>), incl. exceptions
- Message

Example of a log message

```
<Debug
  TimeStamp="2002-10-7T13:44:16.530"
  Host=“te1.hq.eso.org“
  Process="baciTestServer“
  Thread="main"
  Context="“
  File="baciTestClassImpl.cpp“
  Line="205"
  Routine="BaciTestClass::~BaciTestClass
>
  Great debug message!
</Debug>
```

ALMA Project

Log entry type and numerical value

- Trace (1)
- Delouse (2)
- Debug (3)
- Info (4)
- Notice (5)
- Warning (6)
- Error (8)
- Critical (9)
- Alert (10)
- Emergency (11)
- Off (99)

Logging Configuration Overview

- Without anything else, an ACS logger is configured to log TRACE and above levels.
- Optional environment variables can set per-process default log levels.
- The CDB can set per-process log levels and per-logger levels
  - For example, loggers for ACS container and for some stable component log only WARNING and higher, but logger for some unstable component should log DEBUG and higher.
  - Per-logger levels in the CDB “beat” the env vars, while the env vars override the CDB per-process default
- Tools to dynamically change log levels in the running system.

ALMA Project

Logging Configuration Environment Vars

- $**ACS_LOG_STDOUT** sets default values , you can limit which logs are printed to standard out.
  - $ACS_LOG_STDOUT must be between 1 and 11. Lesser values imply that more logs get printed / forwarded.
- Likewise, $**ACS_LOG_CENTRAL** sets the default log level for central logging to the ACS Log service.
- $ACS_LOG_FILE (or default $ACS_TMP/<hostname>/ acs_local_log_<processname>, when not set) - location and root file name for additional stdout-like ACS log files. Most processes generate a unique file name by appending process name and PID to the root file name.

Logging Configuration in the CDB

```
<Container >
  <LoggingConfig
    minLogLevel=“6”
    minLogLevelLocal="8”
    immediateDispatchLevel=“5”
    dispatchPacketSize=“100”
    flushPeriodSeconds="10”  >
     <log:_ Name="jacorb@silentContainer"
        minLogLevel="8" minLogLevelLocal="8" />
  </LoggingConfig>
</Container>
```

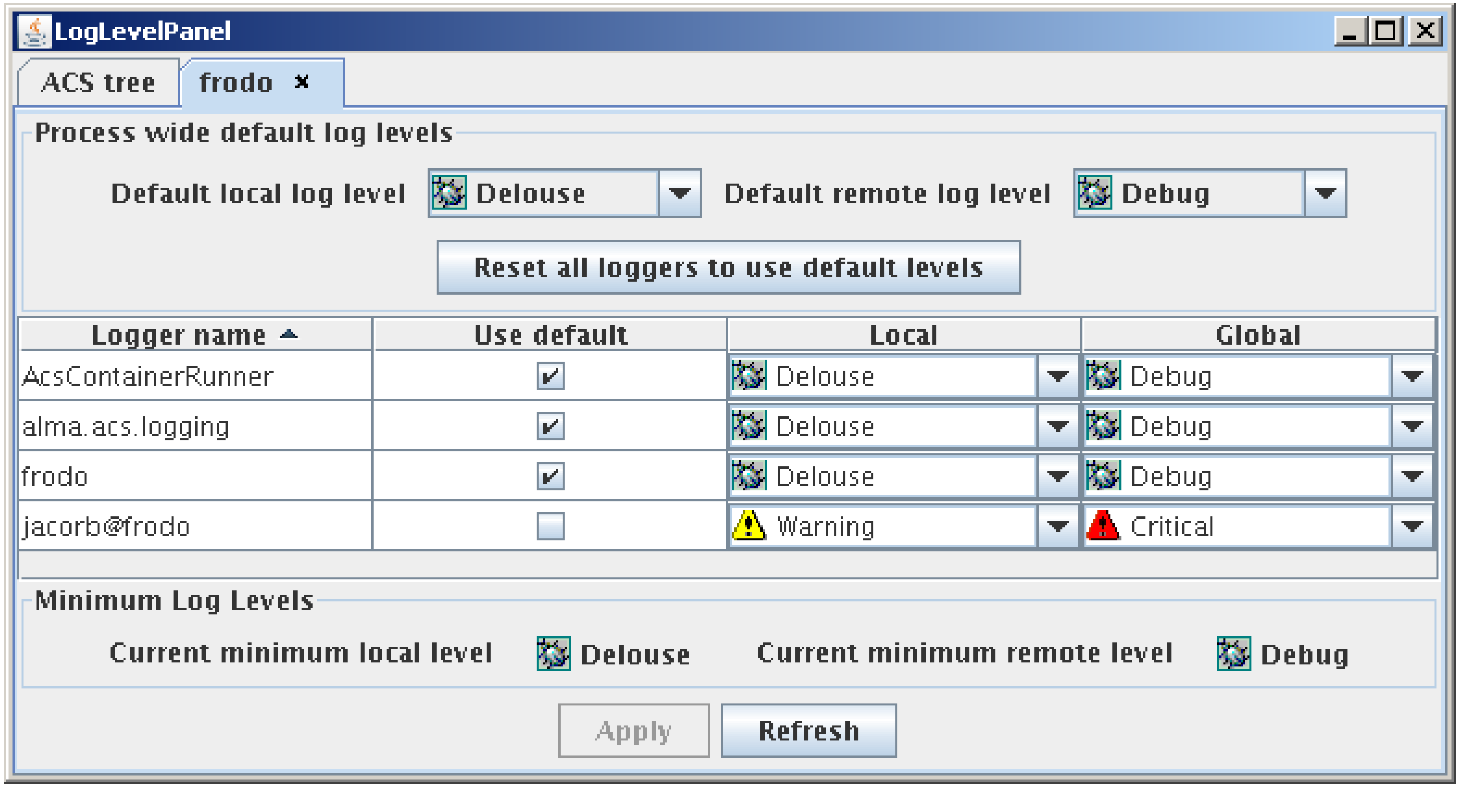
Default log levels

Special levels for a named logger (here: the jacorb logger)

Details are described at http://www.eso.org/projects/alma/develop/acs/OnlineDocs/ACS_docs/schemas/urn_schemas-cosylab-com_LoggingConfig_1.0/complexType/LoggingConfig.html

Logging Configuration At Runtime (2)

GUI “logLevelPanel”



LogLevelPanel

ACS tree | frodo ×

Process wide default log levels

Default local log level: Delouse

Default remote log level: Debug

Reset all loggers to use default levels

Logger name ▲	Use default	Local	Global
AcsContainerRunner	☑	Delouse	Debug
alma.acs.logging	☑	Delouse	Debug
frodo	☑	Delouse	Debug
jacorb@frodo	□	Warning	Critical

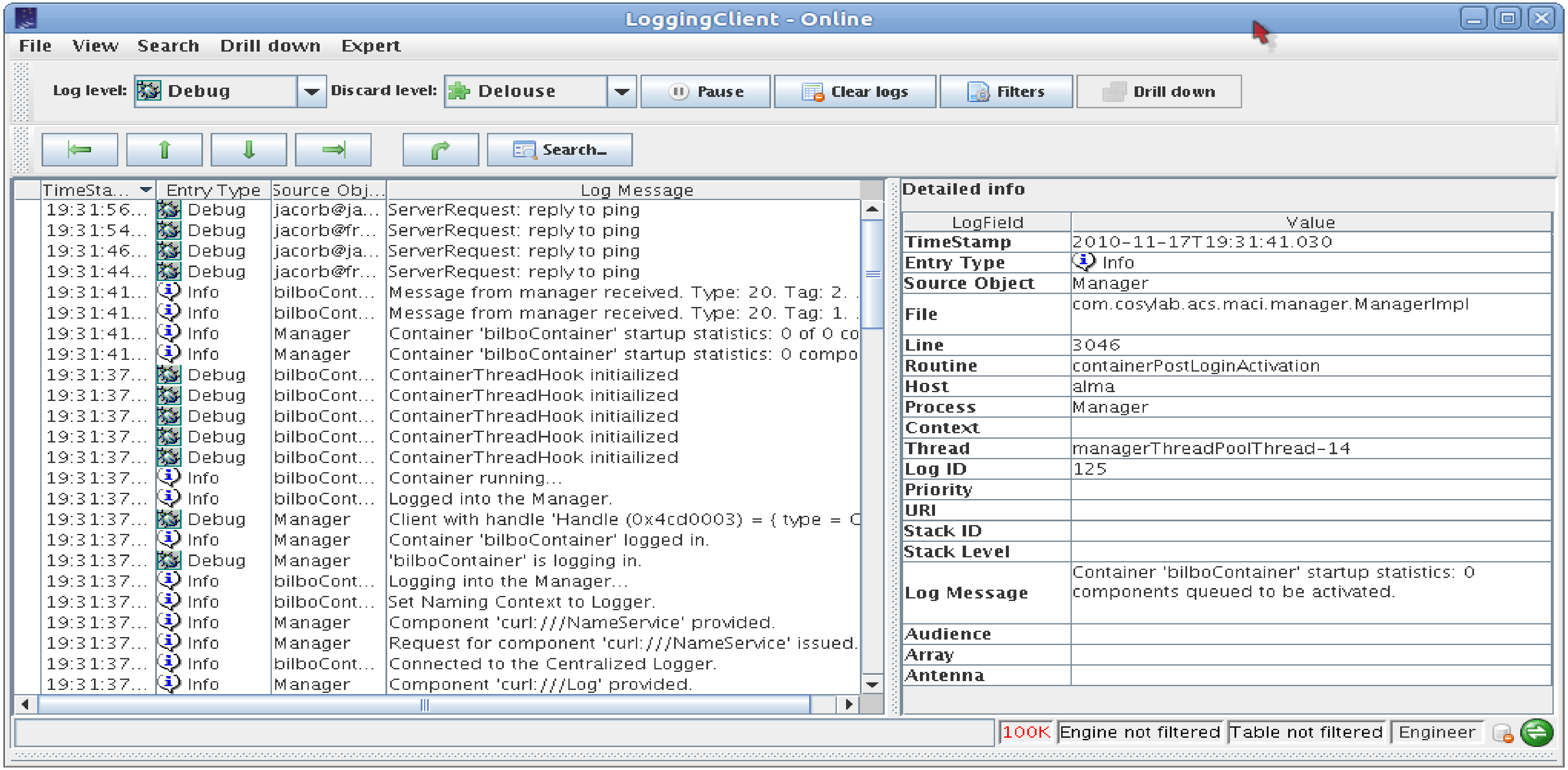
Minimum Log Levels

Current minimum local level: Delouse

Current minimum remote level: Debug

ALMA Project

jlog



TimeSta...	Entry Type	Source Obj...	Log Message
19:31:56...	Debug	jacorb@ja...	ServerRequest: reply to ping
19:31:54...	Debug	jacorb@fr...	ServerRequest: reply to ping
19:31:46...	Debug	jacorb@ja...	ServerRequest: reply to ping
19:31:44...	Debug	jacorb@fr...	ServerRequest: reply to ping
19:31:41...	Info	bilboCont...	Message from manager received. Type: 20. Tag: 2. .
19:31:41...	Info	bilboCont...	Message from manager received. Type: 20. Tag: 1. .
19:31:41...	Info	Manager	Container 'bilboContainer' startup statistics: 0 of 0 co
19:31:41...	Info	Manager	Container 'bilboContainer' startup statistics: 0 compo
19:31:37...	Debug	bilboCont...	ContainerThreadHook initiailized
19:31:37...	Debug	bilboCont...	ContainerThreadHook initiailized
19:31:37...	Debug	bilboCont...	ContainerThreadHook initiailized
19:31:37...	Debug	bilboCont...	ContainerThreadHook initiailized
19:31:37...	Debug	bilboCont...	ContainerThreadHook initiailized
19:31:37...	Info	bilboCont...	Container running...
19:31:37...	Info	bilboCont...	Logged into the Manager.
19:31:37...	Debug	Manager	Client with handle 'Handle (0x4cd0003) = { type = C
19:31:37...	Info	Manager	Container 'bilboContainer' logged in.
19:31:37...	Debug	Manager	'bilboContainer' is logging in.
19:31:37...	Info	bilboCont...	Logging into the Manager...
19:31:37...	Info	bilboCont...	Set Naming Context to Logger.
19:31:37...	Info	Manager	Component 'curl:///NameService' provided.
19:31:37...	Info	Manager	Request for component 'curl:///NameService' issued.
19:31:37...	Info	bilboCont...	Connected to the Centralized Logger.
19:31:37...	Info	Manager	Component 'curl:///Log' provided.

Detailed info

LogField	Value
TimeStamp	2010-11-17T19:31:41.030
Entry Type	Info
Source Object	Manager
File	com.cosylab.acs.maci.manager.ManagerImpl
Line	3046
Routine	containerPostLoginActivation
Host	alma
Process	Manager
Context	
Thread	managerThreadPoolThread-14
Log ID	125
Priority	
URI	
Stack ID	
Stack Level	
Log Message	Container 'bilboContainer' startup statistics: 0 components queued to be activated.
Audience	
Array	
Antenna	

ALMA Project

C++ log messages

- Flags:
  - LM_RUNTIME_CONTEXT
  - LM_SOURCE_INFO
- Log types and default priorities:
  - *LM_TRACE (1)*
  - *LM_DELOUSE(2)*
  - *LM_DEBUG* (3)
  - *LM_INFO (4)*
  - *LM_NOTICE (5)*
  - *LM_WARNING* (6)
  - *LM_ERROR (8)*
  - *LM_CRITICAL (9)*
  - *LM_ALERT (10)*
  - *LM_EMERGENCY* (11)

Java Log Messages

```
m_logger.info("sayHello called...");

catch (Exception ex)
{
    m_logger.severe(“ex in setLampBrightness impl”);
    m_logger.log(Level.SEVERE, "ex in setLampBrightness
   impl", ex);
}
```

```
SEVERE
WARNING
INFO
FINE
FINER
FINEST
```

References

- Logging and Archiving: http://www.eso.org/projects/alma/develop/acs/OnlineDocs/Logging_and_Archiving.pdf
- Documentation for the APIs in the online documentation.