



Maximo Knowledge Transfer Session April 28, 2011

Maximo Logging and Troubleshooting

Manjunath Makonahalli

April 28, 2011

Tivoli software



Agenda

- Logging Overview
- Loggers
- Memory Problems Logging
- SQL Problems Logging
- Database Connection Leak Logging
- TLPA for log analysis



Logging Overview

- Logging is essential to finding problems at run time.
- Without logging and an evidence of where the problem is, it is very difficult to solve the problem.
- Maximo uses Log4j.
 - Came out of open source project and now is part of Java.
- Log4j libraries are packaged as part of Maximo source.
 - Currently using log4j 1.2.13 library.
- Log4j provides the ability to selectively enable or disable logging requests based on the logger.
- Log4j allows logging requests to print to multiple destinations.



Logging Properties

- In Maximo 6, we used logging.properties.
- In Maximo 7, logging properties are moved into a Maximo application called Logging and stored in the database.
 - This is to help configure and apply changes on the fly.
 - Logging.properties is still valid.
 - **If logging.properties exists in the ear file, then the properties file will override the database settings.**
- Let us open a logging.properties and try to understand.
- Three main components to understand
 - LOGGERS
 - LOG APPENDERS
 - LAYOUT



Loggers

- Logger is a category for logging.
 - Named entities, are case sensitive names.
 - SQL logging
 - Database Connection
 - BIRT Reporting
 - WORKORDER application
- Hierarchical – follow hierarchical naming rules.
 - e.g. `log4j.logger.maximo.service.WORKORDER.WOSTATUS=INFO`
- **Named Hierarchy**
 - A logger is said to be an *ancestor* of another logger if its name followed by a dot is a prefix of the *descendant* logger name. A logger is said to be a *parent* of a *child* logger if there are no ancestors between itself and the descendant logger.



Root Loggers

- Under Maximo implementation.
 - Maximo logger is the root logger.
 - The root logger i.e. maximo has to exist, without maximo logger, no logging will work in maximo.
 - To ensure that all loggers can eventually inherit a level, the root logger always has an assigned level.
- Main loggers in Maximo – sometimes called as root loggers.



Root Loggers – Maximo top level loggers

- crontaskmgr log4j.logger.maximo.crontaskmgr
- report log4j.logger.maximo.report
- security log4j.logger.maximo.security
- sql log4j.logger.maximo.sql
- service log4j.logger.maximo.service
- application log4j.logger.maximo.application
- workflow log4j.logger.maximo.workflow
- crontask log4j.logger.maximo.crontask
- datadictionary log4j.logger.maximo.datadictionary
- mail log4j.logger.maximo.mail
- event log4j.logger.maximo.event
- integration log4j.logger.maximo.integration
- exception log4j.logger.maximo.exception // resource intensive
- dm log4j.logger.maximo.dm
- dbconnection log4j.logger.maximo.dbconnection
- interaction log4j.logger.maximo.interaction
- uisession log4j.logger.maximo.webclient.uisession



LOG APPENDERS

- Output destination that stores the log information.
 - Log4j appenders exist for the console, files, GUI components, remote socket servers, JMS, NT Event Loggers, and remote UNIX Syslog daemons.
 - Maximo uses files only.
- Can have multiple log appenders, depending on what is being logged.
- Typical ones
 - Console Appender
 - Rolling File Appender
 - Daily File Appender
 - MX File Appender – Maximo’s extension.
 - MXFileAppender
 - MXDailyFileAppender
 - These basically add server name to the file name. Use “-Dmx.name=<servername>” on the jvm parameters.
- First define an appender, then you can assign the appender to any logger.



Log Appenders - 2

- Each enabled logging request for a given logger will be forwarded to all the appenders in that logger as well as the appenders higher in the hierarchy.
- It is good to have console appender at the root level and specific file appenders are lower levels.
 - `log4j.rootLogger=A1` // root logger appender
 - `log4j.logger.maximo=INFO,A2`
 - `# ~~ Output destinations or appenders`
 - `## A1 is set to be a ConsoleAppender which outputs to System.out.`
`log4j.appender.A1=org.apache.log4j.ConsoleAppender`

 - `# A2 is set to be a RollingFileAppender which outputs to maximo.log file`
 - `log4j.appender.A2=org.apache.log4j.RollingFileAppender`
 - `log4j.appender.A2.File=maximo.log`
 - `log4j.appender.A2.MaxFileSize=5MB`
 - `log4j.appender.A2.MaxBackupIndex=20`



Log Appenders - 3

- Specifying file appenders are lower level.
 - `log4j.logger.maximo.sql=INFO,A3`
 - Where A3 is a log appender defined to hold all sql statements.
 - # A3 is set to be a RollingFileAppender which outputs to `sql_maximo.log` file
 - `log4j.appender.A3=org.apache.log4j.RollingFileAppender`
 - `log4j.appender.A3.File=sql_maximo.log`
 - If you want to collect only the WORKORDER related sql in a file.
 - `log4j.logger.maximo.sql.WORKORDER=INFO,A4`
 - Where A4 is a log appender defined to hold all workorder sql statements.
 - # A4 is set to be a RollingFileAppender which outputs to `workorder_sql_maximo.log` file
 - `log4j.appender.A4=org.apache.log4j.RollingFileAppender`
 - `log4j.appender.A4.File=workorder_sql_maximo.log`



Log Appender Layout

- Layout is used to format the log output.
- This is accomplished by associating a *layout* with an appender.
- Sample
 - `log4j.appender.A1.layout=org.apache.log4j.PatternLayout`
 - `log4j.appender.A1.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss:SSS} [%-2p] %m%n`
- `%d` - date in the format specified inside the brackets.
- `%-2p` – displays the log level
- `%m` – user provided message
- `%n` – platform dependent new line separator
- `16 Apr 2011 16:40:57:636 [INFO] BMXAA7084I - The DbConnectionWatchDog class has been trying to close the database connection for: 73000`
- `%C{1}` - Will print the class name where this message was printed.
- `%M{1}` - Will print the method name where the logging was issued.



Log Appender Layout - 2

- **%M{1}** - Will print the method name where the logging was issued.
- **%t** – Will print the thread info.
- **%%** - will print a single % sign.



Log Levels

- FATAL – Logs on severe errors that indicate application failure
 - Maximo has minimal to none – don't use.
- ERROR – Logs on Errors in the application
 - This is a problem that needs to be looked into.
 - Could be an application problem going back to user or a system level error written to the log file.
- WARN – Logs warnings in the application
 - Not as critical as the ERROR level, but needs to be addressed.
- INFO – Logs informational messages
 - This is mainly for administrators to know what is happening, so they can investigate.
 - Indicates progress inside the application.
- DEBUG – Logs extensive information at very low level of code execution.
 - This is mainly to collect data to be sent to support / development for further analysis.
- TRACE – Further lower level than Debug, Maximo does not use this.



Loggers and Log Level assignment

- Every logger needs a log level.
- If a given logger is not assigned a level, then it inherits one from its closest ancestor with an assigned level.
- **Basic Selection Rule**
 - A log request of level p in a logger with (either assigned or inherited, whichever is appropriate) level q , is enabled if $p \geq q$.
 - Example 1
 - `log4j.logger.maximo.service=ERROR`
 - `log4j.logger.maximo.service.CONFIGURE=INFO`
 - Example 2
 - `log4j.logger.maximo.service=INFO`
 - `log4j.logger.maximo.service.CONFIGURE=ERROR`
- **DEBUG < INFO < WARN < ERROR < FATAL**



Maximo Memory Analysis

- To check whether a maximo server is using excessive memory.
 - Set root logger to INFO level.
 - log4j.logger.maximo=INFO
- You will see messages as below every minute.
 - [4/16/11 16:39:59:933 EDT] 0000003f SystemOut O 16 Apr 2011 16:39:59:933 [INFO] BMXAA7019I - The total memory is 1631584256 and the memory available is 227070992.
- This does not tell you the server maximum memory setting.
 - You need to collect javacore / thread dump information to collect the exact settings on the server.
- When you look through the log and notice that there is very little memory left on the server, then the server is using up lot of memory.
- Next step is to look at the “mbocount” information to see if the problem is with large number of objects in memory.
 - This is done through a system property and not logging. But, the information is written to the log files.



Maximo Memory Analysis - 2

- Set mbocount = 1 through System Properties application.
- This setting produces the following output in log files associated with Maximo root logger.
 - For this to log to a file, the maximo root logger must be at INFO level.
 - 16 Nov 2010 18:09:18:942 [INFO] LOCANCESTOR: mbosets (1168), mbos (1706)
 - 16 Nov 2010 18:09:18:942 [INFO] LOCATIONMETER: mbosets (53), mbos (null)
 - 16 Nov 2010 18:09:18:942 [INFO] LOCATIONS: mbosets (1867), mbos (2934)
 - 16 Nov 2010 18:09:18:942 [INFO] LOCATIONSPEC: mbosets (401), mbos (915)
 - 16 Nov 2010 18:09:18:942 [INFO] LOCCHANGESTATUS: mbosets (4), mbos (null)
 - 16 Nov 2010 18:09:18:942 [INFO] LOCHIERARCHY: mbosets (1945), mbos (2933)
 - 16 Nov 2010 18:09:18:942 [INFO] LOCLEADTIME: mbosets (102), mbos (102)
 - 16 Nov 2010 18:09:18:942 [INFO] LOCOPER: mbosets (557), mbos (1072)
 - 16 Nov 2010 18:09:18:942 [INFO] LOCSTATUS: mbosets (51), mbos (51)
 - 16 Nov 2010 18:09:18:942 [INFO] LOCSYSTEM: mbosets (406), mbos (707)
 - 16 Nov 2010 18:09:18:942 [INFO] LONGDESCRIPTION: mbosets (330), mbos (591)
 - 16 Nov 2010 18:09:18:942 [INFO] MASTERPM: mbosets (2), mbos (2)
 - 16 Nov 2010 18:09:18:958 [INFO] MATRECTRANS: mbosets (17), mbos (17)



SQL Logging

- In Maximo, SQL Logger prints sql statements to the logger specified.
- At root level, the SQL statements are printed for all objects
 - `log4j.logger.maximo.sql=INFO`
- If you want specific object level SQL statements, enable
 - `# ~~ Business Object SQL Loggers`
 - `# log4j.logger.maximo.sql.<service name>.<business object name>=<level>`
 - `# for example:# log4j.logger.maximo.sql.WORKORDER.WOSTATUS=INFO`
- Additionally, the SQL statement time limit logging.
 - Using a system property you can get only the sql statements that exceed a certain time for sql execution.
 - Use `mxe.db.logSQLTimeLimit` property and set to a certain value in milliseconds.



Database Connection Leak Logging and Troubleshooting

- There is a separate logger for database connection leak – Maximo 7113 onwards.
- `log4j.logger.maximo.dbconnection=INFO`
- Enable this and collect log data over a 48 hour time period.
- Will print database connection information every minute.
- Take the latest SystemOut log file.
- Look for the latest set of database connection information towards the end of the file.
 - You only need the print from the last minute as the information is printed every minute.
- Find the connections that are still in use for the longest duration.
- Calculate when the original connection was created from the duration.
- Go to the SystemOut file at that time and search the reference id of the connection.



Database Connection Leak Logging and Troubleshooting

- Db Connection Watchdog prints a stack trace of the first connection acquired by the connection and held for more than 60 seconds.
- If by luck, this is the connection held over time, then you will find the stack trace of the connection creation and you can send the information for a fix.
- If the connection reference printed is not the one held for long, then you have a reference id but it will not be in the logs.
 - This is difficult to trace, until changes are made to the current connection watchdog logging functionality.



BIRT Report Logging

- BIRT reporting has its own logging.
- It is similar to other logging and can be used to log report related problems.

```
log4j.logger.maximo.report  
log4j.logger.maximo.report.birt  
log4j.logger.maximo.report.birt.admin  
log4j.logger.maximo.report.birt.admin.osgibundl  
e  
log4j.logger.maximo.report.birt.queue  
log4j.logger.maximo.report.birt.script  
log4j.logger.maximo.report.birt.sql  
log4j.logger.maximo.report.birt.sql.script  
log4j.logger.maximo.report.birt.viewer
```



Troubleshooting some problems with Logging.

- If you don't see the logs, check the logging setup. In Maximo 7 it is in table called MAXLOGGER.
- You can also ask for an output of the logger setting through the List Logging Properties option.

The screenshot shows the Maximo Logging configuration page. The browser address bar indicates the URL: `http://qawin14.swg.usma.ibm.com:9995/maximo/ui/?event=loadapp&value=logging&uisessionid=22`. The page title is "Logging - Mozilla Firefox IBM Edition".

The main content area displays a table of loggers. The "List Logging Properties" dialog box is open, showing the following logging properties:

```

Logging Properties
log4j.appender.Console-org.apache.log4j.ConsoleAppender
log4j.appender.Console.layout-org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss.SSS} [%-2p] %m%n

log4j.appender.Rolling-psdi.util.logging.MXFileAppender
log4j.appender.Rolling.layout-org.apache.log4j.PatternLayout
log4j.appender.Rolling.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss.SSS} [%-2p] %m%n
log4j.appender.Rolling.File=maximo.log
log4j.appender.Rolling.MaxFileSize=5MB
log4j.appender.Rolling.MaxBackupIndex=20

log4j.appender.DailyRolling-psdi.util.logging.MXDailyFileAppender
log4j.appender.DailyRolling.layout-org.apache.log4j.PatternLayout
log4j.appender.DailyRolling.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss.SSS} [%-2p] %m%n
log4j.appender.DailyRolling.File=maximo_scheduled.log
log4j.appender.DailyRolling.DatePattern=yyyy-MM-dd

log4j.appender.AssetReconTicketWO-psdi.util.logging.MXFileAppender
log4j.appender.AssetReconTicketWO.layout-org.apache.log4j.PatternLayout
log4j.appender.AssetReconTicketWO.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss.SSS} [%-2p] %m%n
log4j.appender.AssetReconTicketWO.File=AssetReconTicketWO.log
  
```

The dialog box has a "Cancel" button at the bottom right.

Done

MAXMESSAGES – Where to look for information

- MAXMESSAGES - Database Configuration contains max messages.
- These are maximo messages that end users see and is also written to log files.
- Contains Error Messages
 - Explanation
 - Operator Response
 - Admin Response
 - System Action



TLPA for log analysis

- <http://tlpa-pc.swg.usma.ibm.com/tlpa/tlpa.jsp>

dfi Report
Items in **RED** should be reviewed closely for resolutions
Click [here](#) to learn how to read and use this report

Log file name: UT1 SystemOut_11.04.19_08.37.57.log
File contains 337250 lines
This log runs from 19 Apr 2011 04:02:57 to [4/19/11 8:37:57:637 EDT]
Database Connect String =

# Of Times	Report Link	Threshold Type / Value	Log Line	Resolutions (if any)
920	SQL Statements Running Longer Than LogSQL Time Limit Report	MAX		Resolutions
7	SQL Running More Than 10 Seconds Report	MAX / 9999	Highest Value Found: [4/19/11 8:23:22:288 EDT] 00000040 SystemOut O 19 Apr 2011 08:23:22:288 [WARN] BMXAA6720W - USER = (CARRIGAD) SPID = (844) app (null) object (WFRANSACTIO) : select * from wfrtransaction where transid = (select max(transid) from wfrtransaction where ownerable='WORKORDER' and ownerid = 2379075 and memo is not null) (execution took 31047 milliseconds) [*** 31 Seconds ***]	Resolutions
275	Total Users Connected Report	MAX	Highest Value Found: [4/19/11 8:35:42:105 EDT] 0000003f SystemOut O 19 Apr 2011 08:35:42:105 [INFO] BMXAA6370I - Total number of users connected to the system: 62	
1273	Users Per JVM Report	MAX	Highest Value Found: [4/19/11 8:35:42:105 EDT] 0000003f SystemOut O 19 Apr 2011 08:35:42:105 [INFO] BMXAA6369I - Server host: 10.250.0.119. Server name: MAXIMOU14. Number of users: 18	
37777	MBOs In Memory Report	MAX	Highest Value Found: [4/19/11 7:53:41:214 EDT] 0000003f SystemOut O 19 Apr 2011 07:53:41:214 [INFO] MAXMENU: mbosets (31), mbos (7045)	
275	Memory Availability Report	MIN		
1	Memory Less Than 10 MB Report	MIN / 10000000	Lowest Value Found: [4/19/11 7:02:40:538 EDT] 0000003f SystemOut O 19 Apr 2011 07:02:40:538 [INFO] BMXAA7019I - The total memory is 1631584256 and the memory available is 8597624.	Resolutions
11321	Long Running Connection Life Times	MAX		
7411	Long Running Connection Life Threshold	MAX / 86400000	Highest Value Found: [4/19/11 8:36:57:637 EDT] Life time:411963779 ms [*** 114 Hours 26 Minutes 3 Seconds ***]	
58	Long Running Connection Exceptions	FIRST	FIRST Instance Found: [4/19/11 4:15:57:630 EDT] psdl.server.DbConnectionWatchDog\$ConnectionData.(DbConnectionWatchDog.java:40)	
39	Users Report			
337250	Full Log			

Warnings were discovered in this analysis - TAKE ACTION TO RESOLVE

Make your tools team better, tells us what you think of TLPA. Click [here](#) to complete a brief survey.



Questions...



Architecture

