How to read Autoform DM logs

Ivana Dordevic - 2024-11-28 - Comments (0) - Autoform DM General Information

Autoform DM

The Autoform DM logs should always be the first place to check for issues if you experience problems with Autoform DM server. To locate the logs, please see the FAQ article 'Where is Autoform DM Installed?'. The logs can be opened with any text editor.

Autoform DM can also be put into Debug logging mode by following the instructions in the article 'How can I put Autoform DM in and out of Debug logging mode?'.

Reading the logs

Autoform DM uses a standard Log4j format:

Log4j format

```
2016-06-13 13:26:30 - (http--0.0.0.0-8100-2   ) -  -
[pdm.userservice.ejb.session.UserAdminSessionFacadeEJB
] DEBUG - Getting UserEntity for username: danielk
```

The log is split into a number of columns and logging levels. The following table shows the details of each:

Column 1	Column 2	Column 3
The time and date stamp. This is the time and date stamp of the server, rather than the client (from which the connections originate). This means that if the client machines are in another time zone, the time may be different in the server logs	Shows where the connection is from, whether it is HTTP or HTTPS, the port used, and the thread ID. The entire string is used as a thread name to allow for someone to follow each request end to end.	After the double dash, the next column shows the category for the logging message, as per the category list that is set out in standalone.xml. By default, logging is set to Info; to change this, look into the node.properties file which has a logging section to set to Debug.

Each category has its own debug settings which can be set in standalone.xml. After the category, the logging level of that particular message is shown. This can be:

- **Trace** - This is the most detailed, not all categories have Trace debugging level available.
- **Debug** - This is the ideal level when investigating issues, this should contain enough detail for debugging most issues.
- **Info** - This is the standard level. It gives very basic information as to what is happening.
- **Warn** - This shows any issues that do not cause a failure of any components.
- **Error** - Error messages only highlight severe issues within DM that need to be investigated.

Only logging messages for the level set in the category or those below it will be output. For example, if Debug is set in standalone.xml, Debug, Info, Warn and Error will output to the log file.

Stack Traces

When an error is encountered when running DM, a stack trace will be output in the log file, which can help identify and resolve the issue. An example of a stack trace is:

Stack Trace

```
2016-06-13 13:27:39 - (http--0.0.0.0-8100-1   ) -  -
[pdm.common.frontend.servlet.DownloadDocumentServlet
] ERROR - Exception in DownloadDocumentServlet: ClientAbortException:
java.net.SocketException: Connection reset by peer: socket write
error
        at
org.apache.catalina.connector.OutputBuffer.realWriteBytes(OutputBuffe
r.java:403)
        at
org.apache.tomcat.util.buf.ByteChunk.flushBuffer(ByteChunk.java:449)
        at
org.apache.tomcat.util.buf.ByteChunk.append(ByteChunk.java:349)
        at
org.apache.catalina.connector.OutputBuffer.writeBytes(OutputBuffer.ja
va:426)
        at
org.apache.catalina.connector.OutputBuffer.write(OutputBuffer.java:41
5)
        at
org.apache.catalina.connector.CoyoteOutputStream.write(CoyoteOutputSt
```

```
ream.java:89)
        at org.apache.commons.io.IOUtils.copyLarge(IOUtils.java:1384)
        at org.apache.commons.io.IOUtils.copy(IOUtils.java:1357)
        at
com.efstech.pdm.common.frontend.servlet.DownloadDocumentServlet.doGet
(DownloadDocumentServlet.java:133)
        at
javax.servlet.http.HttpServlet.service(HttpServlet.java:734)
        at
javax.servlet.http.HttpServlet.service(HttpServlet.java:847)
        at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(Appl
icationFilterChain.java:329)
        at
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationF
ilterChain.java:248)
        at
org.jboss.weld.servlet.ConversationPropagationFilter.doFilter(Convers
ationPropagationFilter.java:62)
        at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(Appl
icationFilterChain.java:280)
        at
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationF
ilterChain.java:248)
        at
com.efstech.pdm.common.frontend.filter.cache.PageExpirationFilter.doH
ttpFilter(PageExpirationFilter.java:39)
        at
com.efstech.pdm.common.frontend.filter.BaseHttpFilter.doFilter(BaseHt
tpFilter.java:28)
        at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(Appl
icationFilterChain.java:280)
        at
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationF
ilterChain.java:248)
        at
com.efstech.pdm.common.frontend.filter.SetCharacterEncodingFilter.doF
ilter(SetCharacterEncodingFilter.java:80)
        at
```

```
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(Appl
icationFilterChain.java:280)
        at
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationF
ilterChain.java:248)
        at
org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperV
alve.java:275)
        at
org.apache.catalina.core.StandardContextValve.invoke(StandardContextV
alve.java:161)
        at
org.jboss.as.jpa.interceptor.WebNonTxEmCloserValve.invoke(WebNonTxEmC
loserValve.java:50)
        at
org.apache.catalina.authenticator.AuthenticatorBase.invoke(Authentica
torBase.java:489)
        at
org.jboss.as.web.security.SecurityContextAssociationValve.invoke(Secu
rityContextAssociationValve.java:153)
        at
org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.j
ava:155)
        at
org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.j
ava:102)
        at
org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineVal
ve.java:109)
        at
org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.jav
a:368)
        at
org.apache.coyote.http11.Http11Processor.process(Http11Processor.java
:877)
        at
org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.proce
ss(Http11Protocol.java:671)
        at
org.apache.tomcat.util.net.JIoEndpoint$Worker.run(JIoEndpoint.java:93
0)
```

```
        at java.lang.Thread.run(Thread.java:722) [rt.jar:1.7.0_07]
Caused by: java.net.SocketException: Connection reset by peer: socket
write error
        at java.net.SocketOutputStream.socketWrite0(Native Method)
[rt.jar:1.7.0_07]
        at
java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:109)
[rt.jar:1.7.0_07]
        at
java.net.SocketOutputStream.write(SocketOutputStream.java:153)
[rt.jar:1.7.0_07]
        at
org.apache.coyote.http11.InternalOutputBuffer.realWriteBytes(Internal
OutputBuffer.java:724)
        at
org.apache.tomcat.util.buf.ByteChunk.flushBuffer(ByteChunk.java:449)
        at
org.apache.tomcat.util.buf.ByteChunk.append(ByteChunk.java:349)
        at
org.apache.coyote.http11.InternalOutputBuffer$OutputStreamOutputBuffe
r.doWrite(InternalOutputBuffer.java:748)
        at
org.apache.coyote.http11.filters.ChunkedOutputFilter.doWrite(ChunkedO
utputFilter.java:126)
        at
org.apache.coyote.http11.InternalOutputBuffer.doWrite(InternalOutputB
uffer.java:559)
        at org.apache.coyote.Response.doWrite(Response.java:594)
        at
org.apache.catalina.connector.OutputBuffer.realWriteBytes(OutputBuffe
r.java:398)
```

This particular stack trace shows a connection issue during the running of the "DownloadDocumentServlet" servlet. When a stack trace occurs in the log file there are a couple of key elements which identify what has happened and where it has happened. At the start of the stack trace, the category can be used to identify the process or engine within DM that has output the error. In this case, it was:

Example

```
pdm.common.frontend.servlet.DownloadDocumentServlet
```

This shows that the error happened on the front-end when trying to download a document.

The second piece of information that should be noted is the end of the Error line. In this case, it was an Exception in DownloadDocumentServlet:

Exception

```
ClientAbortException: java.net.SocketException: Connection reset by peer: socket write error/div>
```

This can be combined by the "Caused by" lined within the stack trace:

Reason

```
Caused by: java.net.SocketException: Connection reset by peer: socket write error
```

In the example, the stack trace information shows that when trying to download a document from the front-end, the user closed the connection before it could be downloaded, causing the error.

There are many different errors that may appear in the logs. They should all have an associated stack trace that can be used to help resolve the issue.