How to enable tracing in OSM 7.3.X

Prior to OSM 7.3.x, an OSM system administrator used the OSM url /OrderManagement/admin/log4jAdmin to change the level of details that OSM can print in (weblogic) server log files. This feature relied on Log4j.

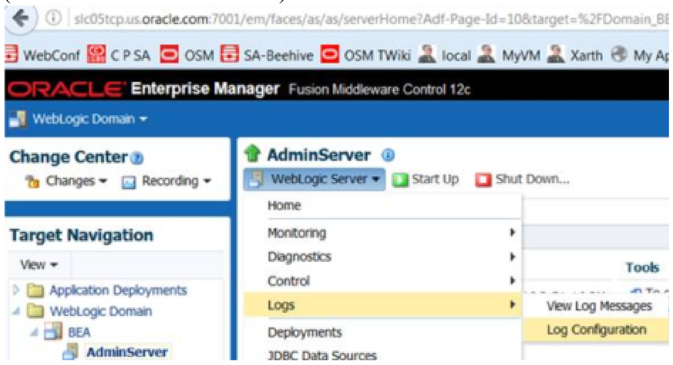
As of 7.3.0 OSM replaced Log4J by ODL (Oracle Diagnostic Logging) which is the standard system logging used by FM applications. The ODL can be viewed and configured using Enterprise Manager.

Typically EM can be accessed in http://AdminServerHost:port/em

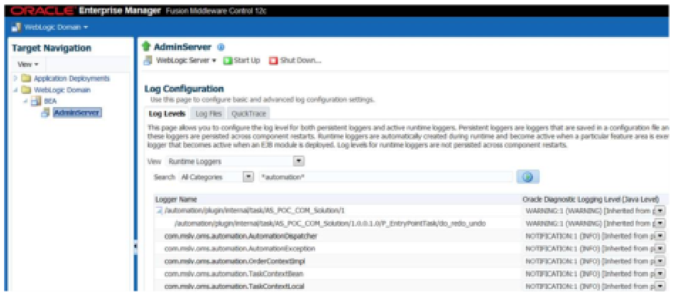
Note that when you enable tracing for OSM, you will find the log entries in the ../logs/**<Managed-Server>-diagnostic.log** file

Steps to Enable Tracing using EM

1. Point web browser to http://AdminServerHost:port/em
2. In left pane expand "Weblogic Domain" and the expand OSM domain
3. Right click the OSM server and Select Logs -> Log Configuration
   Note: The selected server is typically an OSM Managed Server. However in small environments, OSM might be running in the AdminServer (like in the screen shots below)



4. Search Logger by prefix,
   for instance "automations"



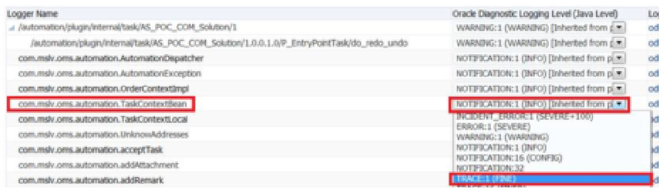
Use this page to configure basic and advanced log configuration settings.

This page allows you to configure the log level for both persistent loggers and active runtime loggers. Persistent loggers are loggers that are saved in a configuration file an these loggers are persisted across component restarts. Runtime loggers are automatically created during runtime and become active when a particular feature area is exe logger that becomes active when an EJB module is deployed. Log levels for runtime loggers are not persisted across component restarts.

View: Runtime Loggers
Search: All Categories "automation"

Logger Name	Oracle Diagnostic Logging Level (Java Level)
/automation/plugin/internal/task/AS_POC_COM_Solution/1	WARNING:1 (WARNING) [Inherited from
/automation/plugin/internal/task/AS_POC_COM_Solution/1.0.0.1.0/P_EntryPointTask/do_redo_undo	WARNING:1 (WARNING) [Inherited from
com.mslv.oms.automation.AutomationDispatcher	NOTIFICATION:1 (INFO) [Inherited from p
com.mslv.oms.automation.AutomationException	NOTIFICATION:1 (INFO) [Inherited from p
com.mslv.oms.automation.OrderContextImpl	NOTIFICATION:1 (INFO) [Inherited from p
com.mslv.oms.automation.TaskContextBean	NOTIFICATION:1 (INFO) [Inherited from p
com.mslv.oms.automation.TaskContextLocal	NOTIFICATION:1 (INFO) [Inherited from p

5. Set the new logging level,
   for instance from INFO to TRACE:1



Logger Name	Oracle Diagnostic Logging Level (Java Level)	Lo
/automation/plugin/internal/task/AS_POC_COM_Solution/1	WARNING:1 (WARNING) [Inherited from	od
/automation/plugin/internal/task/AS_POC_COM_Solution/1.0.0.1.0/P_EntryPointTask/do_redo_undo	WARNING:1 (WARNING) [Inherited from	od
com.mslv.oms.automation.AutomationDispatcher	NOTIFICATION:1 (INFO) [Inherited from p	od
com.mslv.oms.automation.AutomationException	NOTIFICATION:1 (INFO) [Inherited from p	od
com.mslv.oms.automation.OrderContextImpl	NOTIFICATION:1 (INFO) [Inherited from p	od
com.mslv.oms.automation.TaskContextBean	NOTIFICATION:1 (INFO) [Inherited from p	od
com.mslv.oms.automation.TaskContextLocal	INCIDENT_ERROR:1 (SEVERE+100)	od
com.mslv.oms.automation.UnknowAddresses	ERROR:1 (SEVERE)	od
com.mslv.oms.automation.acceptTask	WARNING:1 (WARNING)	od
	NOTIFICATION:1 (INFO)	
com.mslv.oms.automation.addAttachment	NOTIFICATION:16 (CONFIG)	od
	NOTIFICATION:32	
com.mslv.oms.automation.addRemark	TRACE:1 (FINE)	od

Steps to Enable Tracing by modifying the logging.xml

If your test environment does not have EM (very rarely) then you edit the logging.xml under the required server folder

1. Locate and edit the logging.xml file
   ```
   <domain>/config/fmwconfig/servers/AdminServer/logging.xml   ( laptop )
   <domain>/config/fmwconfig/servers/OSM_MS1/logging.xml       ( QA/SIT/PROD )
   ```
2. Change Console level from WARNING to NOTIFICATION to see log.info() in console/stdout
   ```
   <log_handler name='console-handler' class='oracle.core.ojdl.logging.ConsoleHandler' level='NOTIFICATION:32'
   ```

```
formatter='oracle.core.ojdl.weblogic.ConsoleFormatter'/>
```

3. Set AdminServer-diagnostics.log to TRACE:32 level
```
<log_handler name='odl-handler' class='oracle.core.ojdl.logging.ODLHandlerFactory' level='TRACE:32'
filter='oracle.dfw.incident.IncidentDetectionLogFilter'>
```
4. Set OSM logger level to TRACE . For instance: TaskContextBean to debug automation TaskContext
```
<!-- OSM -->
<logger name='com.mslv.oms.automation.TaskContextBean' level='TRACE' useParentHandlers='true'/>
```

Note: There is no need to reboot since logging.xml is constanly monitored by weblogic and automatically re-loaded every time it changes.

Steps to Trace OSM Solution

Typically, solution code written in XQuery can be print logging entries as long as a logger instance is created (or passed by OSM server as context parameter)

1. Assuming the solution has logger instructions like **log.info(), log.debug()**, etc

2. Set Solution automation plug-ins to TRACE level to see OSM developer debug entries coded in xqueries.

   -
     - Logger: `/automation/plugin/internal/task/AS_POC_COM_Solution/1`
     - Level: `TRACE:32`

   or modify **logging.xml** file:
   ```
   <!-- Solution Tasks -->

   <logger name='/automation/plugin/internal/task/AS_POC_COM_Solution/1' level='TRACE:32'
   useParentHandlers='true'/>
   ```

Note: These 2 loggers control tracing for internal and external automation plug-ins:

-
  - **/automation/plugin/internal**/task/AS_POC_COM_Solution/1 will set the logging level for all solution internal automation plug-ins
  - **/automation/plugin/external**/AS_POC_COM_Solution/1 is for all solution external (receiver) automation plug-ins

Note: Search the TaskName as logger name in the ODL Log Configuration page for instance:

Search All Categories: *EntryPointTask* will list logger
like: `/automation/plugin/internal/task/AS_POC_COM_Solution/1.0.0.1.0/P_EntryPointTask/do_redo_undo`

Frequently viewed OSM Loggers

OSM Logger	Results
oracle.communications.ordermanagement.orchestration.generation.AppServerOrchestrationPlanGenerator	• Dumps order *_orchestrationPlanOutput.xml and _orderDataUpdateOuput.xml (ControlData) • Prints out the full path of the 2 xml dumped files. • Prints out OSM Order Header
com.mslv.oms.handler.completeorder.CompleteOrderHandlerEJB	• Dumps PONR_*.xml file when the order is amended
com.mslv.oms.security.base.ControllerBean	• Dumps XML request and response when automation calls context.processXMLRequest() methods

Tracing O2A

O2A has a lot of loggers that can be enabled for tracing.

Some are based on standard logger while other might be based on custom logger (xquery to instantiate a new logger) which print to stdout or custom file under the domain directory.

For instance:

XQuery: OracleComms_OSM_O2A_COM_SalesOrderFulfillment\resources\ComponentInteraction\AIAEBMResponse.xqy

Loggers:

declare variable $logResponseEbm := logfactory:getLog("**OracleComms_OSM_O2A_Response_Ebm**");

declare variable $logResponseDataAfterUpdated:= logfactory:getLog("**OracleComms_OSM_O2A_Response_Data_After_Update**");

declare variable $logResponseOrderLifeCycle := logfactory:getLog("**OracleComms_OSM_O2A_Response_OrderLifeCycle**");

declare variable $logOrderLifeCycleComponentUpdate := logfactory:getLog ("**OracleComms_OSM_O2A_OrderLifeCycle_ComponentUpdate**");

declare variable $logResponseUpdate := logfactory:getLog("**OracleComms_OSM_O2A_Response_Update**");

declare variable $logResponseExtensionDataUpdate := logfactory:getLog ("**OracleComms_OSM_O2A_Response_ExtensionDataUpdate**");

declare variable $logResponseSequencing := logfactory:getLog("**OracleComms_OSM_O2A_Response_Sequencing**");

declare variable $logResponseRedo := logfactory:getLog("**OracleComms_OSM_O2A_Response_Redo**");

declare variable $logResponseUndo := logfactory:getLog("**OracleComms_OSM_O2A_Response_Undo**");

declare variable $logFallout := logfactory:getLog("**OracleComms_OSM_O2A_Fallout**");

Outputs:

• Standard logger	*log:debug(), log:error(), log:info()*
• stdout	*omspiplog:printEle()*
• stdout	*omspiplog:printStr()*
• custom file	*orderlifecyclefn:logOrderActivityToFile()*

Related articles

- OSM 7.3.0 Managing Logs: http://ocinfodev.us.oracle.com/final/osm/73/doc.73/e49157/adm_bea_wls_console.htm#OSMSA694