Tracing in SAP Business Objects Enterprise XI3.1

Applies to:

SAP BusinessObjects Enterprise XI3.0, XI3.1 and other components included in the SAP BusinessObjects Enterprise XI3.x suite. This includes Crystal Reports, Web Intelligence, Desktop Intelligence, Voyager, Dashboard and Analysis, QAAWS, LifeCycle Manager, WDeploy and Enterprise SDK. For more information, visit the Business Objects homepage.

Summary

The purpose of this document is to consolidate the possible tracing options that are available for use in BusinessObjects Enterprise XI3.x. This document was created by using information that is already available in product documentation, Knowledge Base Articles and SDN sources as well as information learned from experience in troubleshooting the XI3.x product.

Author: Brian Thomas

Contributors: James Rapp, Carly Trow, Joshua Kuhn

Company: SAP

Created on: 01 Sept 2011

Author Bio

Brian Thomas' experience with Business Intelligence software began with Seagate Info and Crystal Enterprise back in 2002 with Crystal Decisions. As a senior support engineer, he helped support and troubleshoot large deployments of BusinessObjects Enterprise suite of products. Today, he is part of the SAP Ecosystem team working closely with strategic SAP Partners and their OEM products.

Table of Contents

Introduction

Problem determination in SAP BusinessObjects Enterprise can be at times difficult. The most common method of resolving problems with SAP BusinessObjects Enterprise involves an iterative process of trial and error while generating and analyzing logfiles. Using this method, it is important to understand and isolate which components are used in a given workflow and how a particular problem is reproduced. Once this has been established and specific components are isolated, the involved components can be traced to help uncover the root cause of the issue.

Example:

Issue Synopsis:	*Scheduling a Crystal Report to PDF fails*
Components Used:	*CMS (for scheduling)* *Input/Output FRS (for storing the template and resulting PDF)* *CR Processing Server (for processing the SQL and exporting the report to PDF),*

In the above example, it would be important to further isolate the component in which the failure occurred:

- If the report failed to schedule, or is in “Pending” status, then the CMS log files would be important since it is responsible for scheduling the report.
- If the report was running and failed with the error “Failed to retrieve data from the database”, then the CR Processing Server logs would be important since it is responsible for retrieving data from the database.

This method is the basis for most of the problem determination in SAP BusinessObjects Enterprise.

I urge the reader to use this document as a reference instead of attempting to digest the document as a whole while making use of the table of contents to navigate to a particular section of the document.

Disclaimer

Warning: Most, if not all of the tracing methods described in this document will cause some level of performance impact to the SAP BusinessObjects environment when enabled. Please take this into consideration before performing this type of root-cause determination in a production environment.

Important Note: Always remember to disable tracing once it is no longer required.

BOE Services Trace

SAP BusinessObjects Enterprise has several methods in which services tracing can be enabled. In previous releases, tracing could only be accomplished by stopping a specific service through the Central Configuration Manager (CCM) and adding the directive “-trace” to the command line. The XI3.1 release introduces a new method of tracing services “on-the-fly” without requiring a restart of the services. This allows for improved root cause analysis and problem determination without further disrupting the environment.

Both methods produce similar log files; however the “on-the-fly” _trace.ini trace method is preferred due to its flexibility and ability to be enabled at anytime without requiring the service to be restarted.

Services Trace via <appname>_trace.ini file

Services <appname>_trace.ini is the preferred method of producing trace files from the SAP BusinessObjects Enterprise services and applications. This level of tracing is performed by creating an <appname>_trace.ini file in a specific folder which SAP BusinessObjects services and applications will poll for every 60 seconds. The services or applications that will write trace logs will depend on the actual file name of the .ini file.

The .ini file should contain the following text lines:

```
active = true;
importance = xs;
keep = true;
size = 100000;
```

The following options are available in this .ini file:

Parameter	Values	Description
active	true;false	Enables or disables this tracing. Alternatively deleting or removing the .ini file will disable tracing.
importance	xs; logs lines marked as '<<' and below s; logs lines marked as '<=' and below m; logs lines marked as '==' and below l; logs lines marked as '>=' and below xl; logs lines marked as '>>'	xs, s, m, l, or xl depending on the level you want (xs means everything and xl means only the most serious things).
alert	false, true	Specifies to automatically enable trace for severe system events. Default value is true.
severity	' ', 'W', 'E', 'A', success, warning, error, assert	Specifies the threshold severity over which massages can be traced. Default value is 'E'.
keep	true/false	Configures whether to keep logs after maximum size is reached (default 10,000 lines): True - Create a new log and preserve the previous log False - Create a new log and delete the old log.
size	100000 (recommend 100,000)	Size value indicating number of possible lines written to the logfile. 10,000 lines are about 1.5MB.
log_dir	log_dir="C:/Temp"; example for WINDOWS log_dir="/var/logs"; example for UNIX	This will redirect logfiles to the folder specified. The default is \logging\

Windows services

In order to activate traces on Windows, you have to create a .ini file with a name following the table below and copy it in [BusinessObjects install folder]\BusinessObjects Enterprise 12.0\win32_x86.

Process to monitor	File Name	Specifics
AAAnalytics	AAAnalytics_trace.ini	Server side
AADMining AADMining : XI 3.1 Name **PredictiveAnalysisServer**	AADMining_trace.ini	Server side
AADashboard	AADashboard_trace.ini	Server side
AAMetrics	AAMetrics_trace.ini	Server side
AAProfiler	AAProfiler_trace.ini	Server side
AAQueryMgr	AAQueryMgr_trace.ini	Server side
AARepoMgt	AARepoMgt_trace.ini	Server side
AARules	AARules_trace.ini	Server side
AASPC	AASPC_trace.ini	Server side
AdaptiveProcessingServer	N/A	.ini file tracing is not available for "Java" processes. Alternatives: • In the CMC\Servers - open the AdaptiveProcessingServer and add "-trace" to the Command Line Parameters. • In the CMC\Servers, open the Properties page of the AdaptiveProcessingServer. On the properties page, find the Logging section. In the dropdown menu, change it from "AUTO" to "DEBUG". Other options: "INFO","WARN","ERROR","FATAL"
AdaptiveJobServer	JobServer_trace.ini	Server side
CMS	CMS_trace.ini	Server side
ConnectionServer	ConnectionServer_trace.ini	Server side
EventServer	EventServer_trace.ini	Server side
JobServerChild	JobServerChild_trace.ini	Server side
JobServer	JobServer_trace.ini	Server side
WIReportServer	WIReportServer_trace.ini	Server side
busobj	busobj_trace.ini	Client side (Desktop Intelligence)
crcache	crcache_trace.ini	Server side
crproc	crproc_trace.ini	Server side
crystalras	crystalras_trace.ini	Server side

designer	designer_trace.ini	Client side (Universe Designer)
fccache	fccache_trace.ini	Server side
fcproc	fcproc_trace.ini	Server side
fileserver	fileserver_trace.ini	Server side
Import Wizard	importwiz_trace.ini	Client side (Import Wizard)
Report Conversion/Comparison Tool	javaw_trace.ini	Client side (Report Conversion Tool)
WebIntelligence Rich Client	WebIRichClient_trace.ini	Client side (WebIntelligence Rich Client) • Logs are located in: C:\Documents and Settings\<username>\My Documents\My Business Objects Documents\LocData
Central Configuration Manager	SvcMgr_trace.ini	CCM (cmsdbcopy and updateobjects)

UNIX services

In order to activate traces on UNIX, create with a .ini file with a name following the table below and copy it in [BusinessObjects install folder]\bobje or the current working directory of the command that started your servers.

Process to monitor	**File Name**	**Specifics**
AAAnalytics XI 3.1 Name: DashboardAnalyticsServer	AAAnalytics_trace.ini	Server side
AADMining AADMining : XI 3.1 Name **PredictiveAnalysisServer**	AADMining.bin_trace.ini	Server side
AADashboard	AADashboard_trace.ini	Server side
AAMetrics	AAMetrics_trace.ini	Server side
AAProfiler	AAProfiler_trace.ini	Server side
AAQueryMgr	AAQueryMgr_trace.ini	Server side
AARepoMgt	AARepoMgt_trace.ini	Server side
AARules	AARules_trace.ini	Server side
AASPC	AASPC_trace.ini	Server side
AdaptiveProcessingServer	N/A	.ini file tracing is not available for "Java" processes. Alternatives: • In the CMC\Servers - open the AdaptiveProcessingServer and add

		"-trace" to the Command Line Parameters. • In the CMC\Servers, open the Properties page of the AdaptiveProcessingServer. On the properties page, find the Logging section. In the dropdown menu, change it from "AUTO" to "DEBUG". Other options: "INFO","WARN","ERROR","FATAL"
AdaptiveJobServer	JobServer_trace.ini	Server side
CMS	boe_cmsd_trace.ini	Server side
ConnectionServer	ConnectionServer_trace.ini	Server side
EventServer	boe_eventsd_trace.ini	Server side
JobServerChild	boe_jobcd_trace.ini	Server side
JobServer	boe_jobsd_trace.ini	Server side
WIReportServer	WIReportServer.bin_trace.ini	Server side
crcache	boe_crcached.bin_trace.ini	Server side
crproc	boe_crprocd.bin_trace.ini	Server side
crystalras (not working)	boe_crystalrasd.bin_trace.ini boe_crystalrasd_trace.ini boe_crystalras_trace.ini	Server side
fccache	boe_fccached_trace.ini	Server side
fcproc	boe_fcprocd_trace.ini	Server side
fileserver	boe_filesd_trace.ini	Server side
Central Configuration Manager	mozjsshell_trace.ini	CCM - cmsdbcopy and updateobjects

Client Tools BO_Trace.ini

To trace client side SAP BusinessObjects applications, a single BO_Trace.ini file can enable tracing for all client applications launched.

Warning: This trace is not recommended to be performed on the SAP BusinessObjects Enterprise server. When enabled on the server, tracing is enabled for all services after a reboot. Also, these system environment variables will disable the ability to use the <appname>_trace.ini trace (documented above) until the environment variables are removed and the server is rebooted.

How to enable BO_Trace.ini

1. Create the following system environment variables on the client machine:

   BO_TRACE_CONFIGDIR=C:\LOGS
   BO_TRACE_CONFIGFILE=C:\LOGS\BO_Trace.ini
   BO_TRACE_LOGDIR=C:\LOGS

2. Create a BO_Trace.ini file in the folder specified by BO_TRACE_CONFIGFILE with the following contents:

```
active = true;
importance = xs;
keep = true;
size = 100000;
```

3. Save the file.
4. Launch a client application (The logs will be generated in the directory specified by BO_TRACE_LOGDIR).

Note: if server side tracing is desired, a reboot is required for the services to recognize the new system environment variables. Server side logs are always written to the default \logging\ folder.

When enabled, the BO_Trace.ini method will trace the following client tools and applications:

Application	**Trace File Name**
Business View Manager	CrystalBVM_xx_trace.log CrystalBVM_sdk_xx_trace.log
Central Configuration Manager	SvcMgr_xx_trace.log SvcMgr_sdk_xx_trace.log
Data Source Migration Wizard	BVMigrationWiz_xx_trace.log BVMigrationWiz_sdk_xx_trace.log
Designer	designer_xx_trace.log designer_sdk_xx_trace.log
Desktop Intelligence	busobj_xx_trace.log busobj_sdk_xx_trace.log
Import Wizard	ImportWiz_xx_trace.log ImportWiz_sdk_xx_trace.log
Publishing Wizard	RptPubWiz_xx_trace.log RptPubWiz_sdk_xx_trace.log
Translation Manager	TransMgr_xx_trace.log TransMgr_sdk_xx_trace.log
Universe Builder	designer_xx_trace.log designer_sdk_xx_trace.log
Web Intelligence Rich Client	WebIRichClient_xx_trace.log WebIRichClient_sdk_xx_trace.log
All BOE services traceable via <appname>_trace.ini	[\logging\] folder

Services Trace via <-trace> Command-Line

The **–trace** command-line option is the classic method of generating trace log files from services or applications. It will produce the same logfiles as the <appname>_trace.ini method however, once **–trace** is enabled, some additional command-line options can be included to add even more verbose logging options.

In previous products (Crystal Enterprise 10 or BusinessObjects Enterprise XI Release 2), this trace is enabled by opening the Central Configuration Manager, stopping the service and adding **–trace** to the startup command-line. In SAP BusinessObjects XI 3.1 this is done by logging into the web-based CMC, clicking on "Servers" and editing the command-line by modifying the properties of the particular service.

Note: The –trace directive can also be applied to client applications such as "WebiRichClient.exe" or "ImportWiz.exe" by launching the executable via the "Run" dialogue box with the –trace option or launching via batch file. Be aware that when launched this way, some client applications write logfiles to the user's home directory under \.businessobjects\

The following options are available when using –trace:

Parameter	Possible Values	Description
-trace	off, xs, s, m, l, xl, e, a	Specifies the importance level of messages to trace. By default, even without the –trace directive applied, the trace level applied is 'e' (i.e. only error messages are traced). If set to -trace xs all messages are traced. Use -trace xl if you want to limit tracing to the most serious/critical messages. Default level if unspecified is "xs".
-stackdump		If set to on, a complete stack trace following a critical error is written to the log file. If no value is specified, stackdump is set to on. Unspecified, the default value is off. **Note:** A stackdump will be recorded after any assert is triggered in the logfiles and in some cases asserts are triggered in known-good workflows.
-reqtrace		Records all "Request trace" messages to the log file and the console in debug builds. Request trace messages include request ID information, therefore a single transaction can be traced through the SAP BusinessObjects Enterprise system across several servers. The default value is off.
-noassert		By default, all assert messages are automatically logged. This parameter turns off SASSERT,SASSERTMSG, SVERIFY, and SASSERTMSG_EX messages. Assert messages will not be written to the log file, and will not appear in the console for debug builds.
-nativeassert	off, on	Turns on/off the native assert behavior for the build architecture. The native assert behavior in debug UNIX builds is to dump core. The native assert behavior in debug Windows builds is to pop up a dialog box with diagnostic information.

		Both platforms do not trace assert messages in release builds.
-sdktrace		Specifies whether or not to trace the Business Objects developer libraries. Unspecified, the default value is off.
-filelogfilter	[filename1,filename2]	Sets filters for tracing. To trace messages written to a particular set of files, enter a comma separated list of filenames. To filter out particular files, prefix the comma-separated list of filenames with the ^ character. Example: ^SchedulerSubsystem.cpp,SchedThread.cpp
-loggingPath	[Full directory path]	Specifies the directory where the log file is located. For Windows, the default is the value of the logginPath registry. On UNIX, the default folder is the current working directory - generally the name directory containing the application being traced.
-configFileDir	[Full path to BO_Trace.ini file]	Specifies the directory hosting the <appname>_trace.ini file.
-configFile	<appname>_trace.ini	Specifies the exact location of the <appname>_trace.ini file. Not required. By default, each service will check the \[platform]\ folder.
-loggingplugindir		Specifies the directory containing the logging extension DLLs. For Windows, the default is the value of the logginPluginDir registry. On UNIX, the default folder is the current working directory - generally the same directory containing the application being traced.
-alwaysclose		Specifies if the log file should be closed after a trace is written to the file. The default value is off.
-nevertrace		Specifies to turn off tracing. Once tracing is disables via this option, it cannot be reactivated through the <appname>_trace.ini file. The default value is off.

Trace for other Services and Applications

SIA Trace

The Server Intelligence Agent (SIA) is responsible for maintaining server status as well as executing requests to start, stop, monitor or manage servers on a node.

Name	Possible Values	Description
SIA	-trace “INFO”	• Start the CCM (Central Configuration Manager) • Stop the SIA • Right click on the SIA and click "Properties" • On the "Command:", add the parameter at the end of the line : -trace INFO Eg: -boot "C:\Program Files\Business Objects\BusinessObjects Enterprise 12.0\win32_x86_boe_SIANODE1.bootstrap" -cmspath "C:\Program Files\Business Objects\BusinessObjects Enterprise 12.0\win32_x86\cms.exe" -cmsdir "C:\Program Files\Business Objects\BusinessObjects Enterprise 12.0\win32_x86" -port "6410" -dbinfo "C:\Program Files\Business Objects\BusinessObjects Enterprise 12.0\win32_x86_boe_SIANODE1.dbinfo" -loggingPath "C:\Program Files\Business Objects\BusinessObjects Enterprise 12.0\Logging" -name "SIANODE1" -piddir "C:\Program Files\Business Objects\BusinessObjects Enterprise 12.0\serverpids" -trace "INFO"

Crystal Reports Print Engine (CRPE) trace

Components that run Crystal Reports objects (CR Processing server, CR Jobserver, RAS) utilize the Crystal Reports Print Engine to run the report. CRPEtrace can tell you at a deep level what the engine was doing when running your report.

-crpetrace must be used in conjunction with –trace (example: -trace –crpetrace 7).

Name	Possible Values	Description
CrystalReportsPrintEngine (CRPE)	-crpetrace 1, means trace only CRPE functions -crpetrace 2, means trace only CRPE background thread -crpetrace 3, means trace both CRPE functions and CRPE background thread -crpetrace 4, means trace internal CRPE diagnostics	The CRPE is the engine in which Crystal Reports uses to process SQL and format reports. This type of low-level tracing is only available using –trace –crpetrace 7.

	-crpetrace 7, means trace CRPE functions and CRPE background thread and internal CRPE diagnostics	

Connection Server / Connection Server Libraries trace

The Connection Server and the Connection Server libraries are responsible for connecting and interacting with various report data sources and database middleware. Instead of contacting the Connection Server itself, the WebIntelligence and Desktop Intelligence servers load a ConnectionServer library to handle connectivity to the datasource.

To enable Connection Server / Connection Server library trace:

1. Find cs.cfg file located under the BusinessObjects installation folder:
   ../dataAccess/RDBMS/connectionServer/
2. Edit cs.cfg and locate the traces section:
   <Traces Active="No">
3. Set Active attribute of Traces element to Yes:
   <Traces Active="Yes">
4. Trace the client or server component which implements the connection server libraries by using the "-trace" or .ini trace method described in above

   Example: Enable WIReportServer_trace.ini while cs.cfg is configured for active traces on the WebiReport server.

   Components that can be traced using cs.cfg: WIReportServer, bus_obj, designer, ConnectionServer.

A fully comprehensive guide on tracing the Connection Server, Connection Server libraries and reading Connection Server trace files can be found on the SDN.

SDN Article: Connection Server Troubleshooting Guide

OLAP Data Access Logging

OLAP Datasources can be accessed by several different reporting components in the SAP BusinessObjects suite such as Crystal Reports, Web Intelligence and Voyager.

Data Access Logging is covered in detail in Ruben's SDN article. This article will discuss in detail how to activate tracing to capture information about the data connection and retrieval as well as the timings for each API call (SOFA, SOFA API, MDA, MDA API, Query logging).

SDN Article: OLAP Data Access Logging for Voyager, Web Intelligence, OLAP Intelligence and Crystal Reports

Log4j

Log4j is a Java-based logging utility developed by Apache Software Foundation. Log4j allows for a common platform to enable tracing in SAP BusinessObjects Enterprise for any java-based applications. Log4j can be enabled by editing the corresponding properties file for the java application.

Understanding log4j

Log4j has three main components**: *loggers, appenders*** and ***layouts***. These three types of components work together to enable developers to log messages according to message type and level. These components also help to control at runtime how these messages are formatted and where they are reported.

Loggers

Loggers are defined by the developer in the java code itself. To initialize the logger, the logger will have to be specified in the log4j.properties file.

For example:

log4j.logger.com.businessobjects.rebean is a common logger used for Web Intelligence.

Log4j loggers can be configured to log at different priority levels, which will be described in the following table:

Level	Description	Verbosity
FATAL	Severe errors that cause application termination	Very Small
ERROR	Runtime errors or unexpected conditions	Small
WARN	Runtime situations that are undesirable, but not necessarily wrong.	Medium
INFO	Interesting runtime events.	Large
DEBUG	Detailed information. Most verbose setting	Very Large

Appenders

An appender defines the output destination of a logger. Several types of appenders exist but here we will discuss **console** and **file** appenders.

The console appender will send output to system.out by default. If the application server is running under console mode, it will therefore be printed to the screen.

The file appender can be configured to write messages to a .log file or configured for a **RollingFileAppender** which allows for the logfile to be rolled-over after reaching a maximum size. **RollingFileAppenders** are recommended for most situations.

Note: More than one appender can be attached to a logger.

Layouts

Log4j allows for the user to specify the output format of the logfile according to conversion patterns. For example, the conversion pattern **"%r [%t] %-5p %c - %m%n"** will output something akin to:

176 [main] INFO org.foo.Bar - Located nearest gas station.
Typically, log4j requires a conversion pattern to conform to (example above). SAP BusinessObjects has defined the layout internally inside BOLayout. Throughout SAP BusinessObjects log4j.properties files, you will find:
log4j.appender.REBEAN.layout=com.businessobjects.foundation.logging.log4j.BOLayout
log4j.appender.REBEAN.layout.ConversionPattern=[%c] - %m%n

Troubleshooting

In the event that log4j isn't initializing or working properly, the application server can be configured with the JVM option "**–Dlog4j.debug**" to help troubleshoot.

More information about log4j can be found on Apache's website, or by reviewing the Apache Log4j manual.

External Content: **Apache log4j Manual: Short introduction to log4j**

BOE log4j Tracing

The remaining sections of this document will explain how to enable log4j tracing for some common areas of SAP BusinessObjects Enterprise. Notice that in some cases, it's easiest to simply modify the existing log4j.properties file and in other cases, it's easiest to copy and paste the log4j.properties file contained within the section of this document.

Please note that the level documented will be **DEBUG**, but this can be set to any level as needed. Also remember that this level of tracing can be highly verbose and can contribute to performance issues in a production environment.

Web Intelligence / Desktop Intelligence log4j (REBEAN) Trace

REBEAN trace will log all of the SDK methods utilized in viewing, creating, or modifying a Webi or Deski Report. This is often important in troubleshooting session timeouts, identifying differences between viewers (such as Java vs. HTML), and diagnosing performance issues.

To enable Web Intelligence / Desktop Intelligence (REBEAN) log4j trace:

1. Open the **webi.properties** file located in **Tomcat55\webapps\AnalyticalReporting\WEB-INF\classes**
2. Uncomment the line:
   Trace=1
3. Save the file
4. Rename **log4j.properties** to **log4j.properties.old**
5. Copy the following information into a new file called **log4j.properties**:

```
# Steps to enable REBEAN, WP, CDZLET, DHTML trace
# Set property businessobjects.logs.home to a folder with write permissions

businessobjects.logs.home=C:/Program Files/Business Objects/Tomcat55/logs/

log4j.logger.com.businessobjects.rebean=DEBUG, BO1
log4j.logger.com.businessobjects.wp=DEBUG, BO1
log4j.logger.com.businessobjects.cdzlet=DEBUG, BO1
log4j.logger.com.businessobjects.dhtml=DEBUG, BO1

log4j.appender.BO1=org.apache.log4j.RollingFileAppender
log4j.appender.BO1.File=${businessobjects.logs.home}/AnalyticalReporting.log
log4j.appender.BO1.Append=true
log4j.appender.BO1.MaxBackupIndex=100
log4j.appender.BO1.MaxFileSize=25MB

log4j.appender.BO1.layout=com.businessobjects.foundation.logging.log4j.BOLayout
log4j.appender.BO1.layout.ConversionPattern=[%c] - %m%n
```

6. Save the file
7. Restart Tomcat to enable the trace

Once Tomcat is restarted, a file named **AnalyticalReporting.log** will be created in the folder location specified at the top of log4j.properties.

QAAWS log4j Trace

Query as a Web Service lets you create custom web services for specific queries using Business Objects Web Services. These web services use the "**dswsbobje**" web application and can be used in workflows to retrieve data using LiveOffice or Xcelsius.

To enable QAAWS log4j trace:

1. Open the **webi.properties** file from: **Tomcat\webapps\dswsbobje\WEB-INF\classes**.
2. Uncomment the line:

   Trace=1

3. Save the file.
4. Rename **log4j.properties** to **log4j.properties.old**.
5. Copy the following information into a new file called **log4j.properties**:

```
# Steps to enable DSWSBOBJE trace
# Set property businessobjects.logs.home to a folder with write permissions

businessobjects.logs.home=C:/Program Files/Business Objects/Tomcat55/logs/

log4j.logger.com.businessobjects=DEBUG, DSWSBOBJE
log4j.logger.scripts=DEBUG, DSWSBOBJE
log4j.logger.viewers=DEBUG, DSWSBOBJE
log4j.logger.org.apache.axis=DEBUG, DSWSBOBJE

# Optional – Uncomment to include Webi SDK trace
#log4j.logger.com.businessobjects.rebean.wi=DEBUG, DSWSBOBJE
#log4j.logger.com.businessobjects.wp=DEBUG, DSWSBOBJE
#log4j.logger.com.businessobjects.cdzlet=DEBUG, DSWSBOBJE

# Optional – Uncomment to include TOOLS trace
#log4j.logger.scripts.tools.utils=INFO, DSWSBOBJE
#log4j.logger.scripts.tools.storage=INFO, DSWSBOBJE
#log4j.logger.scripts.tools.settingsManager=INFO, DSWSBOBJE
#log4j.logger.scripts.tools.XMLorHTMLWriter=INFO, DSWSBOBJE

# Optional – Uncomment to include Vintela trace
#log4j.logger.vintela=DEBUG, DSWSBOBJE

# DSWSBOBJE rolling file appender
log4j.appender.DSWSBOBJE=org.apache.log4j.RollingFileAppender
log4j.appender.DSWSBOBJE.File=${businessobjects.logs.home}/DSWSBOBJE.log
log4j.appender.DSWSBOBJE.Append=true
```

```
log4j.appender.DSWSBOBJE.MaxBackupIndex=100
log4j.appender.DSWSBOBJE.MaxFileSize=25MB

# trace format
log4j.appender.DSWSBOBJE.layout=org.apache.log4j.PatternLayout
log4j.appender.DSWSBOBJE.layout.ConversionPattern=%d [%t] %-5p %c{10} (%x) %5r - %m%n
```

6. Save the file.
7. Restart Tomcat to enable the trace.

Once Tomcat is restarted, a file called **DSWSBOBJE.log** will be created in the folder location specified at the top of log4j.properties.

Voyager MDAS log4j Trace

The Multi-Dimensional Analysis Services (MDAS) Server is a java component used for processing Voyager workspaces. The following log4j trace will capture information about the java panel used to create and edit Voyager workspaces.

To enable MDAS trace:

MDAS trace is enabled by default and only the level of trace needs to be modified.

1. Locate

   <BOE_INSTALL_DIR>\BusinessObjects Enterprise 12.0\java\services\MDAS\resources\com\businessobjects\multidimensional\services\mdas.log4j.properties

2. Modify the following line:

   log4j.logger.com.businessobjects.multidimensional=ERROR, file

3. Change ERROR level as desired (FATAL, ERROR, WARN, INFO, DEBUG).
4. Save the file and restart the MDAS service.

This logfile will exist in:
C:/Program Files/Business Objects/BusinessObjects Enterprise 12.0/logging/mdas.log by default (configurable).

Dashboard and Analytics log4j Trace

Dashboard and Analytics applications help you manage and track your company's performance using analytics and dashboards, as well as schedule the refresh of metrics, sets, control charts, Predictive models and analytics.

To enable Dashboard and Analytics log4j trace:

1. Open the **InitConfig.properties** file from: **\Business Objects\Dashboard and Analytics 12.0** folder.
2. Edit the following lines:

   ```
   # Available levels: BASIC < ADVANCED < DETAIL < VERBOSE

       trace.enabled=true
       trace.level=verbose
   ```

3. Save the file.
4. Restart the services.

The appropriate AF.log will be written to the \Business Objects\Dashboard and Analytics 12.0\ folder.

Lifecycle Manager log4j Trace

SAP BusinessObjects LifeCycle Manager (LCM) is a web-based tool that enables you to move BI resources from one system to another system, without affecting the dependencies of these resources. It also enables you to manage different versions of BI resources, manage dependencies of BI resources, and roll back a promoted resource to restore the destination system to its previous state.

To enable Lifecycle Manager log4j trace:

1. Open the log4j.properties file from: **\Tomcat\webapps\LCM\WEB-INF\classes**.

2. Edit the following lines:

```
# # DEBUG < INFO < WARN < ERROR < FATAL
log4j.rootLogger=ERROR, fileAppender
log4j.category.com.businessobjects=ERROR, fileAppender
```

3. Save the file.
4. Restart the services.

The appropriate LCM.log will be written to the \Logging\ folder.

WDeploy log4j Trace

The wdeploy tool is designed to speed up and simplify deployment to Java web application servers.

To enable WDeploy log4j trace:

1. On the machine where you are running WDeploy, locate**: <BOBJE INSTALL ROOT>/deployment/templates/log4j.properties**.

2. Edit the following lines setting the priority and directing to “LogFile” appender:

```
# # DEBUG < INFO < WARN < ERROR < FATAL
log4j.rootLogger=DEBUG, LogFile
log4j.logger.org.apache.tools.ant.Project=DEBUG, LogFile
log4j.logger.org.apache.tools.ant.Target=DEBUG, LogFile
log4j.logger.org.apache.tools.ant.taskdefs=DEBUG, LogFile
log4j.logger.org.apache.tools.ant.taskdefs.Echo=DEBUG, LogFile
```

3. Save the file.
4. Run WDeploy again and he appropriate **wdeploy.log** will be written to the **\deployment\workdir** folder.

BI Platform Java SDK log4j Trace

This trace will help identify any issues that may occur within the SAP BusinessObjects Enterprise Platform Java SDK. This will output some additional information around authentication or session management as well as some common methods used by applications like InfoViewApp.

To enable Enterprise SDK log4j trace:

1. Add the JVM option to the Application Server:

   -Dbobj.logging.log4j.config=verbose.properties

   Or

   -Dbobj.logging.log4j.config=detail.properties

2. Restart the application server.

If the application server is running as a localsystem account, by default the logs will be placed in **\Documents and Settings\Default User\.businessobjects** as either **jce_verbose.log** or **jce_detail.log**.

Related Content

Product Guide: SAP BusinessObjects Enterprise Administrator's Guide

Product Guide: BusinessObjects Voyager Administrator's Guide

Product Guide: Customizing SAP BusinessObjects Web Intelligence with ReportEngine SDK

SDN Article: Connection Server Troubleshooting Guide

SDN Article: OLAP Data Access Logging for Voyager, Web Intelligence, OLAP Intelligence and Crystal Reports

SAP Knowledge Base Article: 1336607 - How to enable log4j logging for Web Intelligence

SAP Knowledge Base Article: 1199027 - How to turn on log4j logging for Query As A Web Service (QAAWS)

SAP Knowledge Base Article: 1197524 - How to enable Performance Manager Recorder Tracing and verbose Log4J output

SAP Knowledge Base Article: 1406756 - How to enable logging (log4j) for LifeCycle Manager (LCM) console

SAP Knowledge Base Article: 1466640 - How to enable verbose logging for WDeploy

For more information, visit the Business Objects homepage.

Copyright