Logs and Logging

How is logging handled?

The Yellowfin & Tomcat log files are extremely useful when troubleshooting issues, or just to find out how things are currently running.

The list of log files (with default options) are listed below:

Log Name	Directory Path	Description
YellowfinInstallLog-YYYYMMDD.log (where YYYYMMDD is the date of installation)	Directly in Yellowfin application folder	This is the installation log file. It logs information about all the chosen installation options, along with any errors encountered during installation.
YellowfinPatchLog–YYYYMMDD.log (where YYYYMMDD is the date the update was run)	Directly in Yellowfin application folder (if an update has been run)	This is the update installation log file. It logs update information (such as updating the database) and will capture any errors encountered. A log file is created each time you run an update.
Yellowfin.log	appserver>logs	This is the Yellowfin application log file. It logs processes/tasks that are run in Yellowfin, such as startup, running reports, exporting items, etc. It will also capture most application errors. By default, this file will cycle once it reaches 1024KB , and will create up to nine files (eg, Yellowfin. log.1, Yellowfin.log.2, and so on, with the most recent data always stored in Yellowfin.log and the oldest in the filename with the largest number). The size and number of files can be adjusted if required. Additional information can be logged by enabling debug logging.
JDBC.log	appserver>logs	This is the Yellowfin configuration database log file. It logs details of the repository database startup and any connection errors.
source.XXXXX.log (Where XXXXX is the ID of the data source)	appserver>logs	These files contain connection information specific to data sources. Each data source has its own ID, so for each data source, a respective log file exists. Note that a log file for a deleted data source will remain until you manually remove it.
catalina.YYYY-MM-DD.log (where YYYY-MM-DD is the date Tomcat was started)	appserver>logs	This is the Tomcat startup log file. It logs any errors encountered while starting the service.
yellowfin-stdout_YYYY-MM-DD.log (where 'yellowfin-' is the name of the Windows service and YYYY-MM-DD is the date Yellowfin was started)	appserver>logs	Note: This file is ONLY present if you have installed Yellowfin as a Windows service. This log file logs information that is usually visible in the console log (the black window that opens when you start Yellowfin).

yellowfin-stderr_YYYY-MM-DD.log (where 'yellowfin-' is the name of the Windows service and YYYY-MM-DD is the date Yellowfin was started)	appserver>logs	Note: This file is ONLY present if you have installed Yellowfin as a Windows service . This log file captures the same errors as the *stdout* log file, but without capturing any other processes.
commons-daemon._service.YYYY-MM-DD.log (where YYYY-MM-DD is the date Yellowfin was started)	appserver>logs	Note: This file is ONLY present if you have installed Yellowfin as a Windows service. This log file logs information relating to the actual Windows service start.
Catalina.out	appserver>logs	Note: This file is ONLY present if you have installed Yellowfin on a Mac (OS X)/Linux box and you're using the 'Startup (background)' option to start Yellowfin. This log file is NOT created on a Mac/Linux box if you are running Yellowfin via the 'Startup (terminal)' option because all info would be logged in the console (as you would see on a Windows box). This log file logs all information relating to the Yellowfin application process; it captures all errors and processes.

Logs and Logging#top

Log files and containerized deployments

The logging method used in the container deployment examples on this wiki all leverage the default logging driver for their respective environment, which by default for Docker and Kubernetes is the **json-file** logging driver. During Yellowfin container uptime, a logging driver writes logs to a JSON file on the container's host. When the container is removed, the logs are removed with it.

For deployments where you wish to view and retain Yellowfin container logs beyond their update — like a production deployment — we recommend shipping the logs to a centralized logging platform using shipping agents that integrate with Docker/Kubernetes.

For more information about logging in Docker and Kubernetes, please refer to Docker Logging Best Practices and Kubernetes Logging Architecture.

Logs and Logging#top

Modifying log files

Log file location

You can modify what information is logged and how log files are treated — including file size, file management and pattern syntax — via the directory *Yellowfin/appserver/webapps/ROOT/WEB-INF/*.

Look for one of the following files:

Yellowfin release	Filename	More info
Yellowfin 9.4 or later	log4j2.xml	Logging has been upgraded to Log4j2 and works differently in Yellowfin 9.4 and beyond. See https://logging.apache.org/log4j/2.x/index.html for further details.
Yellowfin 9.0–9.3	log4j.properties	Earlier versions of Yellowfin used Log4j logging, which has reached end of life. See http://logging.apache.org/log4j/1.2/ for further details.

Before making any changes to the file, we recommend that you back up your existing file and place it in a different location.

If you wish to email these files, you may need to stop the Yellowfin service (especially on Windows) as it will either not allow you to send, or send blank files.

Appenders and loggers

Appenders and loggers work together to deliver log events to log files. An appender defines a pattern for each log line, the destination of the line (file, DB, cloud etc.) and any associated configuration. A logger tells Yellowfin what to log and level of logging. A logger also maps Java class files to the appender that the class should use to write its logs. A number of Log4j2 appenders and loggers already exist and are detailed on the log4j2 appenders page.

Enable RepositoryErrorAppender

RepositoryErrorAppender is a custom Log4j appender for saving error log messages to the Events table in the Yellowfin repository database.
Error messages having lesser severity such as INFO or DEBUG are not written. However, more severe error messages, such as ERROR or FATAL, are written to the database.

By default, RespositoryErrorAppender is not used. If you wish to enable the appender, you must first add the appender and then associate it with the logger category or root category within your log config file, which is described below.

Yellowfin 9.4 and later

To enable the appender in Yellowfin 9.4 or later, follow the steps below.

1. Open your **log4j2.xml** file (see above for more information on its location)
2. Find the `<Appenders>` element and add a new line:

```
<RepositoryErrorAppender name="repos"/>
```

3. Associate the appender with the logger level or root category in the `<Loggers>` element
   In the example below, we've added the line with "repos" to Root: Root is the default logger; it has no parents and is therefore at the top of the log4j2 logger hierarchy. It therefore logs all error messages from anywhere in Yellowfin's backend to the repository DB:

```
<Root level="INFO">
    <AppenderRef ref="applog"/>
    <AppenderRef ref="cons" />
    <AppenderRef ref="repos" />
</Root>
```

4. If you prefer, you can also associate the appender with an entire group of classes
   In the example below, we've added the line with "repos" to the class group of `com.hof.cluster`, which logs all error events from Yellowfin's clustering classes to the repository DB: this is irrespective of the `INFO` level defined for this category because the code in this particular appender allows only ERROR or higher to be logged:

```
<Logger name="com.hof.cluster" level="INFO">
    <AppenderRef ref="repos" />
</Logger>
```

Yellowfin 9.0–9.3

To enable the appender in Yellowfin 9.0-9.3, follow the steps below.

1. Add the following line to the **log4j.properties** file (see above for more information on its location):

```
log4j.appender.repos=com.hof.adapter.RepositoryErrorAppender
```

2. Associate the appender with a logging category or the root category of the log4j.properties file
   In the example below, we've added "repos" to the following line for `rootCategory`, so it logs all error messages from anywhere in Yellowfin's backend to the repository DB:

```
log4j.rootCategory=INFO, cons, applog, repos
```

3. If you prefer, you can also associate the appender with an entire group of classes
   In the example below, we've added the line with "repos" to the class group of `com.hof.cluster`, which logs all error events from Yellowfin's clustering classes to the repository DB: this is irrespective of the `INFO` level defined for this category because the code in this particular appender allows only ERROR or higher to be logged:

```
log4j.category.com.hof.cluster=INFO, repos
```

Edit the sourcelog appender

The sourcelog appender of Log4j logs messages about each source database in their own individual log files.
This appender creates a RollingFileAppender for each source and supports most properties that the RollingFileAppender supports.
The "file" parameter has a %s token, which is automatically replaced with the Yellowfin sourceId.

NOTE: this logger is intended for use by the Yellowfin DBConnectionPool. While the appender may be configured, there is little value in using the appender for any other logger.

By default, the sourcelog appender is enabled.

Yellowfin 9.4 and later

If you wish to modify the sourcelog appender, it can be found in the <Appenders> element area of your **log4j2.xml** file.

```
<SourceLog name="sourcelog" fileName="C:/logs/source.%s.log" filePattern="C:/logs/source.%s.log.%i"
maxFileSize="1024KB" maxFiles="9">
<PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSS} %6p: %m%n" />
</SourceLog>
```

Yellowfin 9.0–9.3

If you wish to alter the sourcelog appender, look for the following lines in your **log4j.properties** file:

```
log4j.appender.sourcelog=com.hof.pool.SourceLogAppender
log4j.appender.sourcelog.File=C:/logs/source.%s.log
log4j.appender.sourcelog.MaxFileSize=1024KB
log4j.appender.sourcelog.MaxBackupIndex=9
log4j.appender.sourcelog.layout=org.apache.log4j.PatternLayout
log4j.appender.sourcelog.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %6p: %m%n
```

Logs and Logging#top

Section navigation

Current topic - Deploy Yellowfin

This page is part of the Install And Deploy Yellowfin section of the wiki, which has these topics:

Install on Premises

Logs and Logging

- Installation Steps

Install in the Cloud

Install in the Cloud

- Yellowfin for AWS
- Yellowfin for Azure
- Yellowfin for Google Cloud Platform

Install in a container

Install in a Container

- Docker
- Kubernetes
- Upgrading Yellowfin Container Deployment

Deploy Yellowfin

Deploy Yellowfin

- Logs and Logging
- Yellowfin Directory Structure
- User Welcome

Advanced Deployments

Advanced Deployments

- Clustering Guide
- Yellowfin Server Specification
- Automate Yellowfin Deployment on Linux
- SAML Bridge
- Standalone Configuration Tools

top