Logging

Overview

The JFrog Platform provides standardized logs for all JFrog products and their services. All logs include a standard format and naming convention.

This page describes the different available logs, their location in the system directory and how they should be used.

> **Additional References**
>
> For more information, see the System Architecture and System Directories pages.

Page contents

Log Files Location and Naming

For each JFrog service you will find its active log files in the `$JFROG_HOME/<product>/var/log` directory. For consistency, each log file is prefixed by its service name and a dash, `<service-name>-service.log`. For example, `artifactory-service.log` and `router-request.log`.

The following log files are included for all JFrog Services:

Service Log	`<service-name>-service.log` For example: `artifactory-service.log`	Main service log file for each microservice, containing data on the service activity.
Request Log	`<service-name>-request.log` For example: `artifactory-request.log`	Lists all http requests (including gRPC) that were made to the service.
Outbound Request Log	`<service-name>-request-out.log` For example: `artifactory-request-out.log`	Lists all the remote requests initiated by a remote repository and replication.
Console Log	*console.log*	Combined log file that contains server activity for all microservices. In Windows, microservices will have separate console log files named *<service-name>-console.log*

Archived Logs

Each log file has default rolling policy which will compress the log file and move it to the `$JFROG_HOME/<product>/var/log/archived` folder.

Log File Structure

The Request and Access log files each display specific type of activity and as such have a consistent and specific file structure for maximum readability

Service Log

The service log file console pattern uses colors to highlight the service type and message level. On Windows console colors should be disabled.

Service log file record structure

```
Timestamp (UTC) [Service Type] [Level] [Trace Id] [Class and Line Number] [Thread] - Message
```

Service log file record sample

```
2018-11-18T15:39:04.902Z [jfac ] [INFO ]  [4b1b8a0b04e31b80] [s.r.NodeRegistryServiceImpl:44] [http-exec-
4          ] - request to "join" with serviceId jffe@000
```

Value	Description	Example
Timestamp	The date and time the message was logged, in UTC time with the standard format: [yyyy-MM-dd'T'HH:mm:ss.SSSZ] based on RFC-3339	`2018-11-18T15:39:04.902Z`
Service Type	The service type, color coordinated with a specific color for each service, including: • Artifactory: Bright Green • Access: Yellow • Event: Bright Cyan • Router: Cyan • Tomcat: Magenta • Metadata: Bright Blue • Xray: Yellow Cross product services (such as router, tomcat, scripts) use the same color.	`[jfrpg]`

Level	The service identifier as a 4 to 6 character long, including:	`[jfrt ]`

JFrog Product	Service Name	Service ID
Artifactory	Artifactory	`jfrt` (legacy: `jf-artifactory`)
	Access	`jfac` (legacy: `jf-access`)
	Router	`jfrou`
	Metadata	`jfmd`
	Frontend	`jffe`
	Event	`jfevt`
	Replicator	`jfrep`
	JFLink	`jfcon`
	Mission Control	`jfmc`
	Integration	`jfint`
	Observability	`jfob`
Xray	Server	`jfxr`
	Analysis	`jfxana`
	Indexer	`jfxidx`
	Persist	`jfxpst`
	Indexer-App	`jfxia`
Distribution	Distribution	`jfds`
	Distributor	`jfdr`
Mission Control (Below version 4.7)	Mission Control	`jfmc`
	Insight Server	`jfisv`
	Insight Schedular	`jfisc`
Pipelines	extensionsync	`jfpes`
	Logup	`jfplog`
	Marshaller	`jfpmar`
	Hook Handler	`jfphh`
	Nexec	`jfpnex`
	Cron	`jfpcrn`
	Step Trigger	`jfpst`
	Run Trigger	`jfprt`
	Pipeline Sync	`jfpps`
	Template Sync	`jfpts`
	Request Sealer	`jfprs`
	Frontend	`jfpwww`
	Api	`jfpapi`
	Pipelines router	`jfprou`
Installer	Installers Commons	`jfin`

Trace Id	The trace id value. Trace id is used to identify a request across services	`4b1b8a0b04e31b80`
Class and Line Number	The fully qualified class name and line number printing this log entry.	`s.r. NodeRegistryServiceI mpl:44`
Thread	The thread printing this log entry. "main" if not java.	`[http-exec- 4 ]`
Message	The log entry message.	`Hello JFrog`

Request Log

The request log file pattern contains a list of pipe ("|") separated values. The file pattern will contain the same number of columns, if a value is missing it will be empty.

Note: If not provided by the client, the '`Request Content-Length`' value is initialised as "`-1`".

Request log file record structure

```
Timestamp | Trace ID | Remote Address | Username | Request method | Request URL | Return Status | Request Content Length | Response Content Length | Request Duration | Request User Agent
```

Request log file record sample

```
2018-11-18T15:39:04.902Z|d5d75b3c41242768|127.0.0.1|anonymous|GET|api/v1/cert/root|200|0|6|0|JFrog Access Java Client/4.1.12
```

Value	Description	Example
Timestamp	The date and time the request was completed and entered into the log file, in UTC time with the standard format: [yyyy-MM-dd'T'HH:mm:ss.SSSZ].	`2018-11-18T15:39: 04.902Z`
Trace ID	The trace id value.	`4b1b8a0b04e31b80`
Remote Address	The IP address of the remote caller (ipv4 or ipv6).	`10.0.12.3`
Username	The requesting user's username or "anonymous" when accessed anonymously.	`benn`
Request method	The HTTP request method, in UPPERCASE.	`GET, PUT`
Request URL	The relative URL for the request.	`api/v1/cert/root`
Return Status	The HTTP return code for the request.	`201`
Response Content Length	The size of the server response in bytes, for example, the size of downloaded file. -1 if unknown (for example, chunked encoding).	
Request Content Length	The size of the user request in bytes, for example, the size of an uploaded file. -1 if unknown.	
Request Duration	The time in ms for the request to process.	
Request User Agent	The request user agent.	`JFrog Access Java Client/4.1.12`

Outbound Request Log

The `request-out log` file pattern contains a list of pipe ("|") separated values. The file pattern will contain the same number of columns, if a value is missing it will be empty.

Note: If not provided by the client, the '`Request Content-Length`' value is initialised as "`-1`".

Request log file record structure

```
Timestamp | Trace ID | Remote Repository Name | Username | Request method | Request URL | Return Status |
Request Content Length | Response Content Length | Request Duration
```

Request log file record sample

```
2021-05-12T13:58:46.686Z|40ea218a769325db|generic-remote|andreyt|HEAD|https://acme.jfrog.com/artifactory
/generic-packages/jdbc-drivers/mssql-jdbc-7.4.1.jre11.jar|200|1219373|0|80
```

Value	Description	Example
Timestamp	The date and time the request was completed and entered into the log file, in UTC time with the standard format: [yyyy-MM-dd'T'HH:mm:ss.SSSZ].	`2018-11-18T15:39:04.902Z`
Trace ID	The trace id value.	`4b1b8a0b04e31b80`
Remote Repository Name	The name of the remote repository.	`generic-remote`
Username	The requesting user's username or "anonymous" when accessed anonymously.	`benn`
Request method	The HTTP request method, in UPPERCASE.	`GET, PUT`
Remote URL	The URL for the remote resource.	`https://acme.jfrog.com/artifactory/generic-packages/jdbc-drivers/mssql-jdbc-7.4.1.jre11.jar`
Return Status	The HTTP return code for the request.	201
Response Content Length	The size of the server response in bytes, for example, the size of downloaded file. -1 if unknown (for example, chunked encoding).	
Request Content Length	The size of the user request in bytes, for example, the size of an uploaded file. -1 if unknown.	
Request Duration	The time in ms for the request to process.	

Router Request Log

The JFrog Router has a JSON based access log containing all the requests that went through the Router, including service service communication.

Below is an example of an entry in the Router request log (router-request.log)

Router Request Log Entry


```
{
  "BackendAddr": "http://localhost:8049",
  "ClientAddr": "127.0.0.1:61899",
  "DownstreamContentSize": 2,
  "DownstreamStatus": 200,
  "Duration": 8353000,
  "RequestMethod": "GET",
  "RequestPath": "/router/api/v1/system/ping",
  "StartUTC": "2020-11-12T11:53:03.605300906Z",
  "request_Uber-Trace-Id": "4ccb40200c199346:1a3f95ce1b27711d:71e15f8b6031c9e9:0",
  "request_User-Agent": "curl/7.54.0",
  "time": "2019-08-05T14:42:09+03:00",
  "level": "info",
  "msg": ""
}
```


Value	Description	Example
BackendAddr	Address of the backend server the request was forwarded to	`http://localhost:8049`
ClientAddr	The IP address of the remote caller in its original form (ipv4 or ipv6, usually IP:port).	`127.0.0.1:61899`
DownstreamContentSize	The number of bytes in the response entity returned to the client.	`2`
DownstreamStatus	The HTTP return code for the request.	`200`
Duration	The time in nanoseconds for the request to process.	`8353000`
RequestMethod	The HTTP request method, in UPPERCASE.	`GET`
RequestPath	The relative URL for the request.	`/router/api/v1/system/ping`
StartUTC	The date and time request processing has started, in UTC time with the standard format: [yyyy-MM-dd'T'HH:mm:ss.SSSSSSSSSZ].	`2020-11-12T11:53:03.605300906Z`
request_Uber-Trace-Id	The full trace id value.	`4ccb40200c199346:1a3f95ce1b27711d:71e15f8b6031c9e9:0`
request_User-Agent	The request user agent.	`curl/7.54.0`
time	The date and time the request was completed and entered into the log file, in UTC time with the standard format: [yyyy-MM-dd'T'HH:mm:ss.SSSZ]	`2019-08-05T14:42:09+03:00`
time / msg	Default info and empty message	

Console Log

The console log file appends the console outputs of all services into one common log file.

Log rotation is configured to occur every hour using a cron job for Docker Compose and native installations.

Log rotation is not available in the following installations:

1. Archive
2. Mac/Windows
3. Manual Docker Compose (which don't use the bundled script)

Since this file is written to by all services and can grow quickly, it is recommended to manage it by either by disabling it using the `shared.logging.consoleLog.enabled` configuration in the Artifactory System YAML, or by setting up your own log rotation.

You have to configure log rotation manually for Tomcat logs. For more information, see Configuring Log Rotation for Tomcat.

Viewing Log Files from the UI

You can view essential Platform log files from the UI.

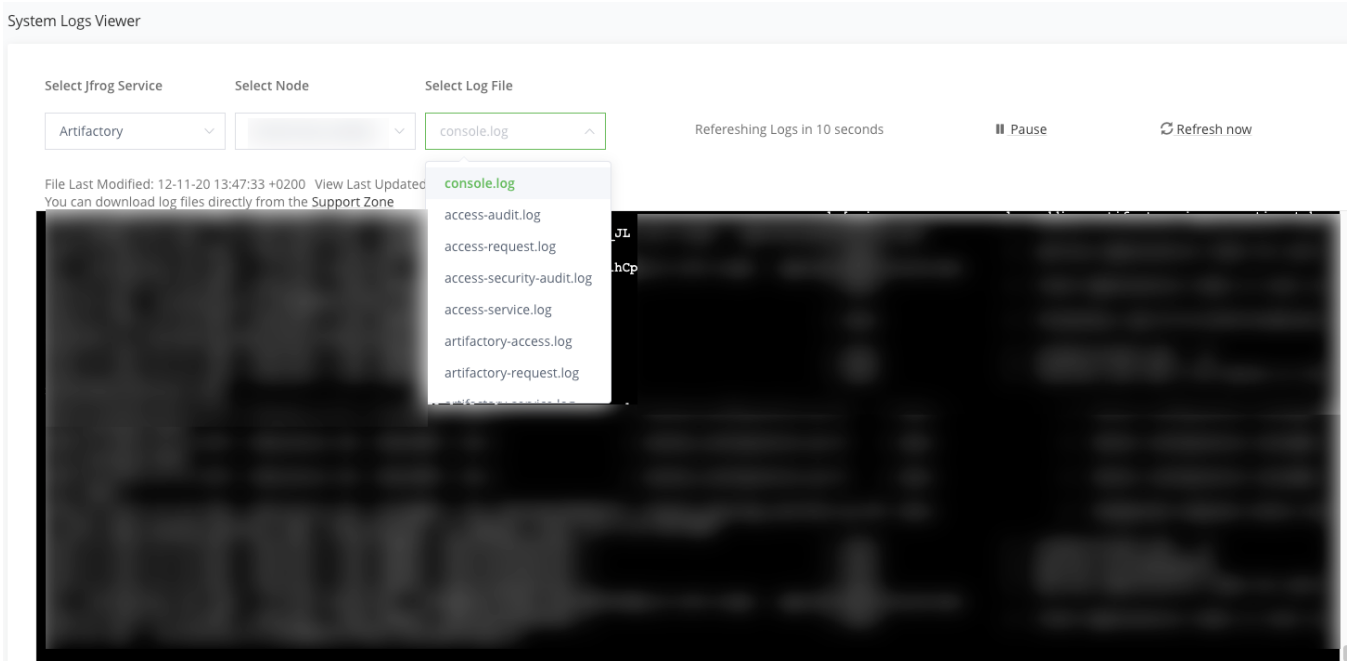
Important Details

This feature is supported on a JFrog Self-Hosted solution only.

To view system logs:

1. In the **Administration** module, go to **Monitoring | System Logs.**
2. Select the JFrog service you want to view logs for.
3. Select the node.
4. Select the file you want to view.
   The log tail view is automatically refreshed every few seconds, however can be paused and resumed if you wish to browse the log.

To save system resources, do not leave the log view open in your browser unnecessarily.



Sending Logs to Syslog

Some sites want to consolidate logs into the syslog facility. The following steps will enable you to send your Java microservices logs to syslog.

Configure the logback library

Edit the logback xml file in the `$JFROG_HOME/<product>/var/etc/<microservice>/logback.xml` file. For example, to configure Artifactory to use syslog, edit the `$JFROG_HOME/artifactory/var/etc/artifactory/logback.xml` file.

1. Add the following syslog appender to the logback xml (next to the other appenders)

```
<appender name="SYSLOG" class= "ch.qos.logback.classic.net.SyslogAppender">
 <syslogHost>localhost</syslogHost>
 <facility>SYSLOG</facility>
 <suffixPattern>[%thread] %logger %msg</suffixPattern>
</appender>
```

2. Add the following appender to the output:

```
<root>
 <level value="warn"/>
 <appender-ref ref="CONSOLE"/>
 <appender-ref ref="FILE"/>
 <appender-ref ref="SYSLOG"/>
</root>
```

3. Save the file, and restart the service.

Configure syslog on your machine

Since logback is using internet sockets, you have to make sure your syslog facility accepts them. Modern Linux distributions are using the rsyslog daemon for syslogging. Ensure that the configuration for internet domain sockets is enabled, either by editing `/etc/rsyslog.conf` and uncommenting:

```
# Provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
# Provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514
```

or placing it in a file under `/etc/rsyslog.d` ending in `.conf`.

Restart rsyslog.

```
service rsyslog restart
```

Configuring Log Verbosity

There are two ways to configure log verbosity, depending on if your JFrog microservice is logback based (Java microservices) or not.

Using logback (Java based microservices)

The verbosity of any Java based logger in your system can be configured by entering or modifying the level value in the corresponding entry in the Logback configuration file `JFROG_HOME/<product>/var/etc/<microservice>/logback.xml`. For example, to configure the Artifactory log verbosity, edit the `$JFROG_HOME/artifactory/var/etc/artifactory/logback.xml` file.

Changes made to the logging configuration are reloaded within several seconds without requiring a restart.

Modifying the verbosity of a logger in logback.xml

```
<logger name="org.artifactory.http.out" level="debug"/>
```

Using system.yaml (non Java microservices)

The verbosity of any non Java based logger in your system can be configured by entering or modifying the level value in the corresponding entry in the `system.yaml` configuration file `JFROG_HOME/<product>/var/etc/system.yaml`.

Changes made to the logging configuration requires a restart.

Modifying the verbosity of a logger in system.yaml

```
frontend:
  logging:
    application:
      level: info
```