



CHAPTER 7

CONFIGURATION OF RECORD SUBSYSTEM

General description	
Meta	Configure log handlers and loggers. •
Goals	Configure different types of record handlers. • Configure ROOT and category loggers. •
sections	Configuration of record handlers (and exercise guided) • Configuration of loggers (and guided
Laboratory work	exercise) • Configuration of the logging subsystem

Configuring record handlers

Goals

After completing this section, students should be able to do the following:

- Configure different types of record handlers.

Log subsystem overview

Registration is one of the most important tasks an application server performs. Collecting and analyzing log files is an important task for an EAP administrator to help diagnose, troubleshoot, and analyze issues in production. Sometimes there may also be a business requirement to store application registration information to comply with regulatory and compliance laws.

In keeping with the modular design of EAP 6, a subsystem handles registration tasks. The registry configuration is presented in the *registry subsystem* and is configured in the configuration files `domain.xml` (in managed domain mode) and `standalone.xml` (on a standalone server).

The EAP registration subsystem consists of three main components:

1. **Handlers:** A handler determines "where" and "how" an event will be logged. EAP comes out of the box with several handlers that can be used to log application messages.
2. **Loggers (loggers):** A logger (also called a log category) organizes events into logically related categories. A category is typically assigned to a given package namespace, running inside the Java virtual machine (can be EAP as well as custom application package names) and defines one or more handlers that will process and log messages. .
3. **Root Logger (root logger):** Loggers are organized in a hierarchy of elements parent-child and the root logger resides at the top of the logger hierarchy.

A handler is activated only if one or more loggers refer to it. Loggers can reference more than one handler (or none at all). The root logger captures all log messages of the specified log level that were not captured by a log category.

By default, the EAP logging level is INFO and two handlers are enabled:

- **CONSOLE:** a handler that logs events in the console window.
- **FILE:** A handler that logs events to a file named `server.log`.

Default log file locations

Previous versions of EAP had a single log folder. EAP 7 has multiple log file locations depending on whether EAP is running as a standalone server or in managed domain mode.

For example, if EAP is running as a standalone server, the log files are located at:

```
$BASE_DIR/standalone/log
```

where `BASE_DIR` is the EAP base folder. The `standalone/log` folder contains two log files:

- **gc.log.0.current** – For logging the Java virtual machine flags with which EAP started and memory management events while the server is running. The values for the settings in this file are in the `BASE_DIR/bin/standalone.sh` file. The `gc.log.0.current` file shows only log events that occurred during the most recent EAP boot.
- **server.log** – For server log events. Log events are appended to this file; therefore, typically the server will need to replace this file based on file size and time (for example, hourly or daily). The values for the `server.log` configuration are in the logging subsystem section of `standalone.xml`.

If EAP is operating in domain managed mode, registration occurs in multiple locations (relative to the EAP base folder):

- **domain/log/host-controller.log** – Contains log events that occur during the start of the host controller (or domain controller in case this particular host is the domain controller). It also logs events such as lost connection to the domain controller or newly registered host controllers, as well as debugging or runtime errors on the host controller.
- **domain/log/process-controller.log** – Contains log events related to starting and stopping processes on the host. (Note that the host controller runs in its own process, and remember that each server on the host also runs in its own process.)
- **domain/servers/<server-name>/log/server.log** – Contains log events for this server.

Note that each server on a host has its own folder for log files.

For example, if a host has a server named `dev-server-one`, the log file for that server is as follows:

```
domain/servers/dev-server-one/log/server.log
```

The `<server-name>/log/server.log` file can be compared to a standalone server's `server.log` file and is a good starting point when searching for

Chapter 7. Configuring the registry subsystem

problems on a server This file is appended each time the server starts, so it is recommended to rotate `server.log` once an hour or day, or when the file reaches a certain size. Next, we'll discuss configuring log files, and configuring a size-based rotating log file in the guided exercise later in this chapter.



use

You can change the default location of the log folders by setting the `jboss.server.log.dir` property at runtime. Remember that the `jboss.server.log.dir` folder is relative to `jboss.domain.base.dir`, so if `jboss.domain.base.dir` is defined, a different value for `jboss.server.log.dir` is not needed. , unless there are specific requirements for the managed domain.

built-in handlers

EAP includes six built-in handlers. The built-in handlers are as follows:

- **Console Handler:** defines a handler that writes to the console. Is set by using the `<console-handler>` element.
- **File Handler:** defines a handler that writes to a file. Is set by using the `<file-handler>` element.
- **Periodic Rotation File Handler:** Defines a handler that writes to a file and rotates the record after a period derived from the given suffix. It is configured by using the `<periodic-rotating-file-handler>` element.
- **File handler that rotates based on size:** defines a handler that writes to a file and rotates the log once the file size grows past a certain point, and maintains a fixed number of backups. It is configured by using the `<size-rotating-file-handler>` element.
- **Asynchronous Handler:** defines a handler that writes to subhandlers in a asynchronous thread. This is useful in scenarios where the application generates a large number of log messages in parallel threads and where file system I/O throughput becomes a bottleneck. It is configured by using the `<async-handler>` element.
- **syslog handler:** defines a handler that sends events to a remote syslog server. It is configured by using the `<syslog-handler>` element.



use

Handlers specify where and how an event is logged, but the definition of a handler does not automatically trigger them. Activation is done in the `<logger>` or `<root-logger>` tags, which we'll talk about later in this chapter.



use

A custom handler can be developed for logging events to any resource type and in any required format. This can be accomplished by writing a Java class that extends the base EAP registration framework API, and then defining a custom-handler in the EAP configuration file. Writing a custom handler is beyond the scope of this course.

Handlers are configured in the logging subsystem section of the server's configuration file. For example, standalone.xml and most of the profiles in domain.xml come already preconfigured with a `<console-handler>` defined. The CLI definition is similar to the following:

```
{
  "outcome" => "success",
  "result" =>
    { "autoflush" => true,
      "enabled" => true,
      "encoding" => undefined, "filter"
      => undefined, "filter-spec" =>
      undefined, "formatter" =>
      "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n", "level" => "INFO", "name"
      => "CONSOLE", "named-
      formatter" => "COLOR-
      PATTERN", "target" => "System.out"
    }
}
```

The corresponding XML looks like this:

```
<subsystem xmlns="urn:jboss:domain:logging:3.0">
  <console-handler name="CONSOLE" autoflush="true">
    <level name="INFO"/>
    <formatter>
      <named-formatter name="COLOR-PATTERN"/>
    </formatter> </
  <console-handler>

  ...

</subsystem>
```

- The `<level>` element specifies the minimum logging level for this handler.
- The `<formatter>` element allows you to specify a named pattern to use when logging events. The `COLOR-PATTERN` formatter paints the log output for different levels and makes it easy to detect errors when analyzing large log files. For example: Log messages with severity level `ERROR` or `FATAL` are painted red to draw the administrator's attention when viewing the logs.

The `CONSOLE` handler can also be configured using the EAP management console in the Configuration > Subsystems > Registry section. Click Registry, select the Handler tab, and then click the Console link:

Chapter 7. Configuring the registry subsystem

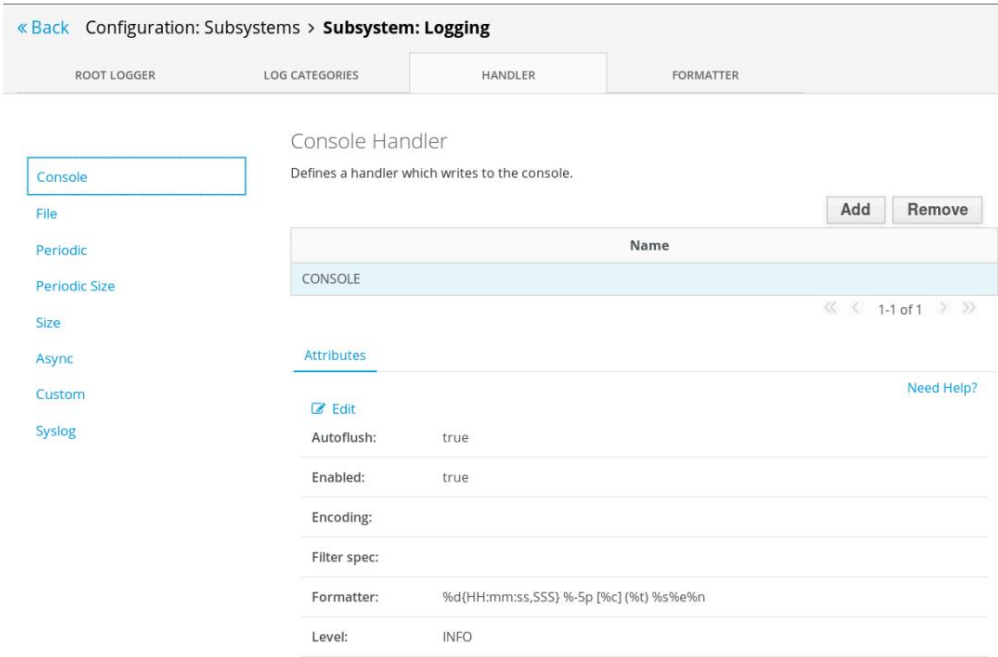


Figure 7.1: Console manager

File handlers that rotate periodically

Use periodic-rotating-file-handler to automatically rotate log files based on a certain elapsed time. For example, the default FILE handler is configured to replace the server.log file every 24 hours. The definition of the handler is as follows:

```
<periodic-rotating-file-handler name="FILE" autoflush="true"> <formatter>
    <named-formatter name="PATTERN"/>
</formatter>
<file relative-to="jboss.server.log.dir" path="server.log"/> <suffix value=".yyyy-MM-
dd"/> <append value="true"/>
</periodic-rotating-file-handler>
```

- The name of this handler is FILE and it is the handler that EAP uses regularly. default.
- The suffix value determines how often the log file is rotated. In it example above, the suffix is yyyy-MM-dd, which represents a day format. Therefore, this log will rotate once per day (at midnight).
- The value of relative-to is a path element (defined in this configuration file) or a Java property. In this case, we are using the predefined property jboss.server.log.dir, which points to the BASE_DIR/standalone/log folder on a standalone server.

File handlers that rotate periodically

The suffix value is a date and time format, and the pattern specifies how often the log file rotates. For example, if a log file should rotate once an hour, set the suffix to:

```
yyyy-MM-dd.HH
```

Similarly, monthly rotation can be achieved by setting the suffix to:

```
yyyy-MM
```

The periodic-rotating-file-handler can be configured and managed using the EAP administration console in the Configuration > Subsystems > Registry section. Click Registry, the Handler tab, and then the Periodic link to view, edit, or add a periodic-rotating-file-handler:

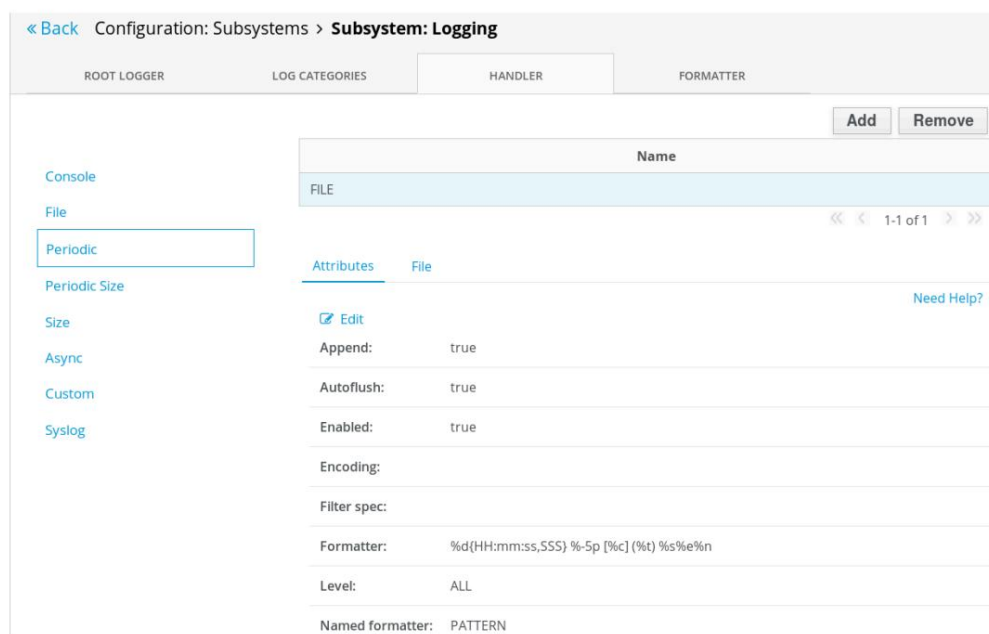


Figure 7.2: File manager that rotates periodically

You can also use the EAP CLI to define a new handler by using the add operation. For example, the following CLI command adds a new periodic rotating-file-handler with the name MY_HANDLER:

```
/profile=default/subsystem=logging/periodic-rotating-file-handler=MY_HANDLER:add( autoflush=true, append=true,
named-

formatter=PATTERN,

file={ path=mylog.log, relative-
      to=jboss.server.log.dir
    },
level = INFO ,
suffix =

)
```

Chapter 7. Configuring the registry subsystem

Size-rotating file handlers Use size-rotating-file-handler to automatically rotate log files based

on log file size. This handler is useful in scenarios where the number of log files generated in a period is very large and can lead to performance bottlenecks if the log file is many gigabytes in size. The handler can be configured to rotate the log file after a certain size (rotate-size value). Below is a sample definition of this handler:

```
<size-rotating-file-handler name="FILE_BY_SIZE" > 1
  <level name="INFO" /> 2
  <formatter>
    <named-formatter name="PATTERN" /> 3
  </formatter>
  <file relative-to="jboss.server.log.dir" path="production-server.log" /> 4 5
  <rotate-size value="1m" /> 6
  <max-backup-index value="3" /> 7
</size-rotating-file-handler>
```

- 1 The name attribute is required and is the name of the handler. Must be unique within a profile (in domain managed mode).
- 2 The level attribute sets the default logging level for this handler. The handler manages all log messages at this level and further down the logger hierarchy.
- 3 The formatter attribute references a valid named-formatter reference defined in the configuration file. The pattern can also be provided directly by using the pattern-formatter tag and defining a pattern template.
- 4 The relative-to attribute refers to the full path where the log file should be created, or it can refer to a logical path reference defined in the configuration file. The default value is the predefined property `jboss.server.log.dir`, which points to the `BASE_DIR/standalone/log` folder on a standalone server and the `BASE_DIR/domain/servers/<server-name>/log` folder in domain managed mode .
- 5 The path attribute provides a path to the log file, which is relative to the value of the relative-to path. For example, if the path value `mylog.log` is given, the log file is created at `BASE_DIR/standalone/log/mylog.log` on a standalone server and at `BASE_DIR/domain/servers/<server-name>/log /mylog.log` in domain managed mode.
- 6 The rotate-size value is the size of the log file after which the handler rotates the file. The substituted file is appended with a suffix with a numerical index. The rotation size value can be given in megabytes by suffixing the value with an "m". For example, to rotate the log file after 5 MB, the rotate-size value should be given as 5m.
- 7 The max-backup-index attribute determines how many backup copies of the rotated log files are kept before the files are reused. For example, for a log file named `mylog.log` with a max-backup-index value of three, the handler will rotate the log file three

Demo: Configuring file handles that rotate based on size

times (after it reaches the rotate-size value) in log files named mylog.log.1, mylog.log.2, and mylog.log.3. On the next rotation, mylog.log will be overwritten and old log messages will be removed. Make sure that the max-backup-index value is set appropriately, based on the rate at which the application generates logs.

File handles that rotate based on size can be defined in the management console in the Configuration > Subsystems > Registry section. Click Registry and then the Size link on the Handler tab.

The instructor will teach how to define a file handle that rotates based on size using the EAP admin console:

Demo: Configuring file handles that rotate based on size

Review the video to perform the steps later. Play it again as many times as you deem necessary.

1. Open a terminal window from the workstation virtual machine (Applications > Favorites > Terminal) and run the following command to create a lab directory and verify that EAP is installed and is not currently running:

```
[student@workstation ~]$ demo logging-srfh setup
```

2. Users often want to be able to run multiple standalone instances without needing to install EAP multiple times on the same host. In the previous guided exercise, an alternate directory was created to customize the EAP instance and leave the original installation intact.

Verify that you can see the following three folders inside the /home/student/JB248/labs/standalone folder:

- configuration
- deployments
- lib

3. The jboss.server.base.dir variable allows users to submit a directory path to use as an alternate base directory for server content. Run the following command to start the EAP server using the standalone.sh script in the original EAP installation, while using the new EAP configuration files:

```
[student@workstation standalone]$ cd /opt/jboss-eap-7.0/bin [student@workstation bin]$ ./standalone.sh \ -Djboss.server.base.dir=/home/student/JB248/labs/standalone/
```

The server should start without issue with the following message:

Chapter 7. Configuring the registry subsystem

```
17:03:30,497 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: JBoss EAP 7.0.0.GA (WildFly Core 2.1.2.Final-redhat-1) started in 3313ms -Started 261 of 509 services (332 services are lazy, passive or on-demand)
```

4. On the workstation, navigate to `http://localhost:9990` to access the console EAP administration.

Log in with the username `jbossadm` and the password `JBoss@RedHat123`.

5. In the EAP management console, the Logging subsystem is configured in the Configuration > Subsystems > Logging section.

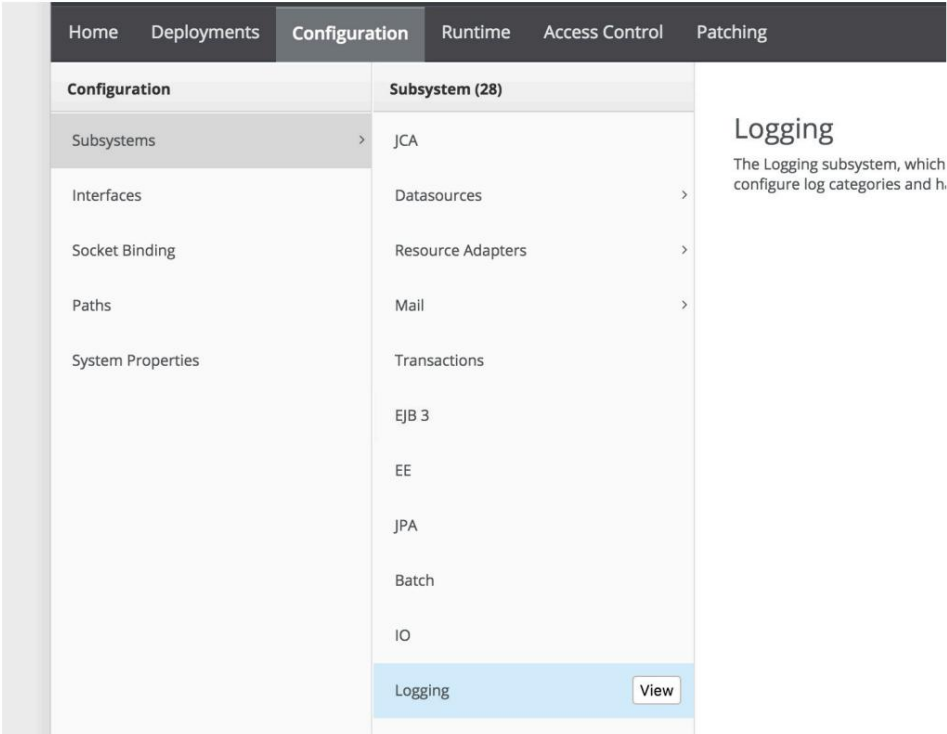


Figure 7.3: Registration subsystem

Navigate to Configuration > Subsystems > Logging in the EAP management console and click View next to Logging to display the logging subsystem configuration page.

6. On the logging subsystem configuration page, there are four tabs available at the top of the menu that offer the following configuration options:

- **ROOT LOGGER:** The default logging settings used by EAP and all the applications and subsystems implemented.
- **LOG CATEGORIES:** a filter used to obtain logs from certain applications/ subsystems.

Demo: Configuring file handles that rotate based on size

- **HANDLER:** defines where and how the log files are generated and the configurations that EAP and all applications/subsystems can use.
- **FORMATTER:** configures the result generated by a handler.

7. Click the **HANDLER** tab on the logging subsystem configuration page, and then click the **Size** link on the left bar menu.

« Back Configuration: Subsystems > Subsystem: Logging

ROOT LOGGER LOG CATEGORIES HANDLER FORMATTER

Size Handler

Defines a handler which writes to a file, rotating the log after the size of the file grows beyond a certain point and keeping a fixed number of backups.

Add Remove

Name
No Items!

« < > »

Attributes File

Need Help?

Append:

Autoflush:

Enabled:

Encoding:

Filter spec:

Figure 7.4: File manager that rotates based on size

8. Click **Add** on the size handler configuration page to add a new handler for files that rotate based on size.
9. In the popup window that appears, enter **FILE_BY_SIZE** as the value in the **Name** field and click **Next**. Note that the handler name must be unique, as there may be multiple handlers defined in the EAP configuration file.

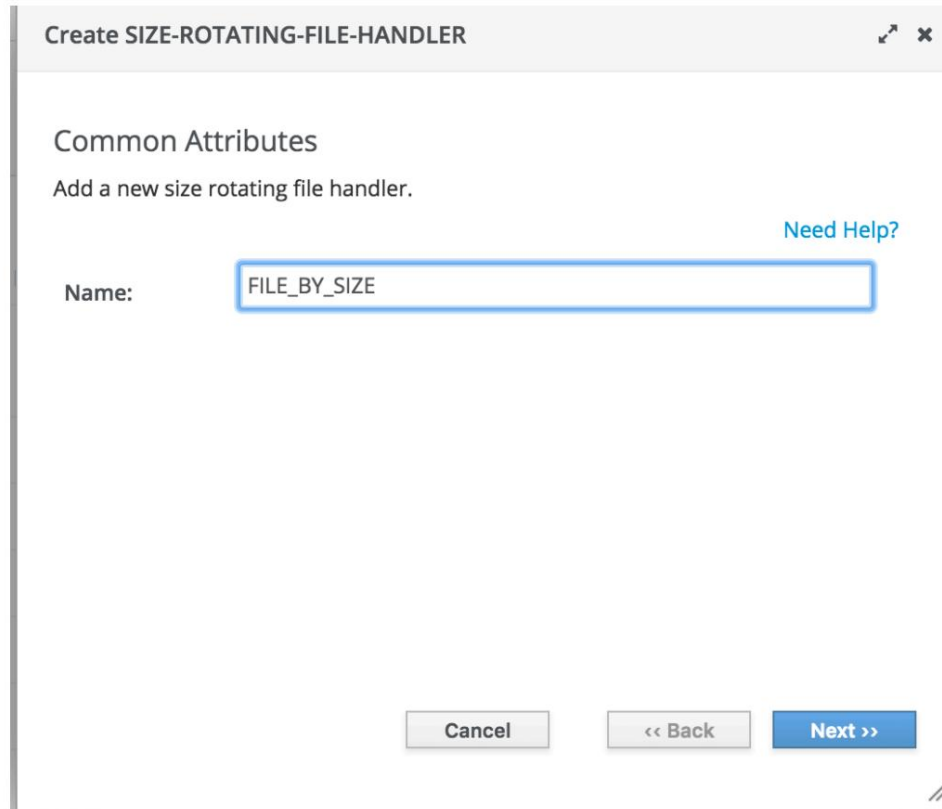


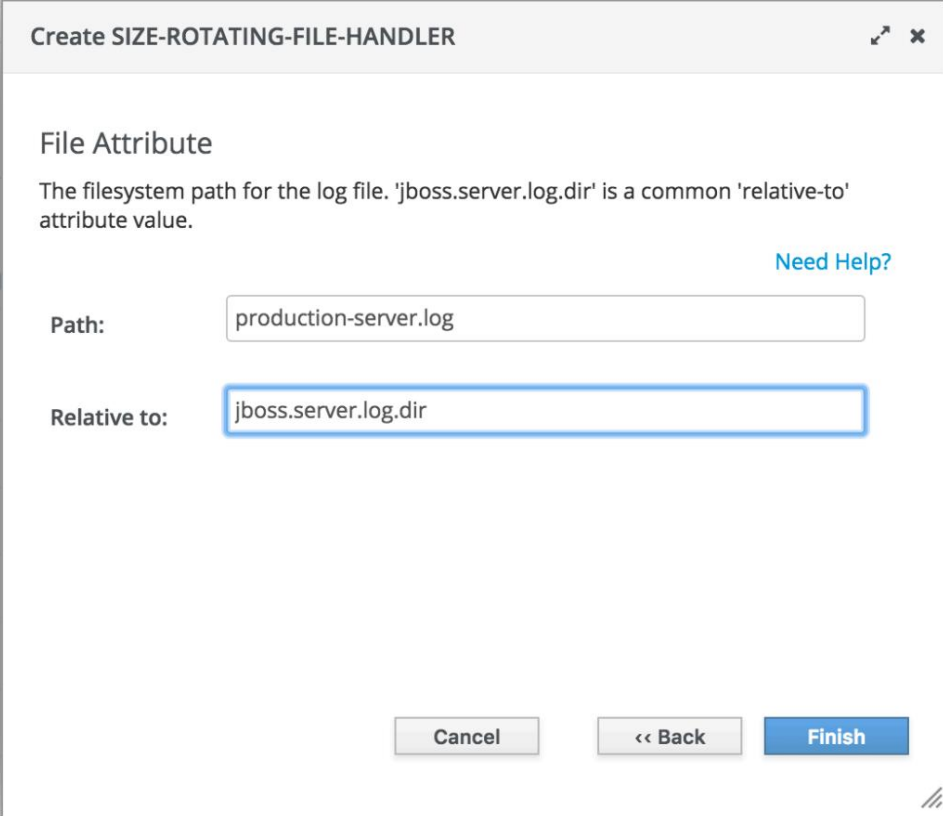
Figure 7.5: Adding a new handler for files that rotate based on size

On the next page, enter `production-server.log` as the value in the Path field and `jboss.server.log.dir` as the value in the Relative path field. Finally, click Finish to create the handler.



use

`jboss.server.log.dir` is an EAP-defined system variable, which points to `${jboss.server.base.dir}/log` by default.

Demo: Configuring file handles that rotate based on size

Create SIZE-ROTATING-FILE-HANDLER

File Attribute

The filesystem path for the log file, 'jboss.server.log.dir' is a common 'relative-to' attribute value.

[Need Help?](#)

Path: production-server.log

Relative to: jboss.server.log.dir

Cancel << Back Finish

Figure 7.6: Configuring file handle attributes that rotate based on size

You should now see a new handle for rotating files based on size called **FILE_BY_SIZE**, which appears on the size handler configuration page. Notice that the Level field is set to **ALL** and the Rotation Size field is set to **2m (2MB)**. Setting the level to **ALL** will return log messages at all levels from all loggers implemented in EAP.

Chapter 7. Configuring the registry subsystem

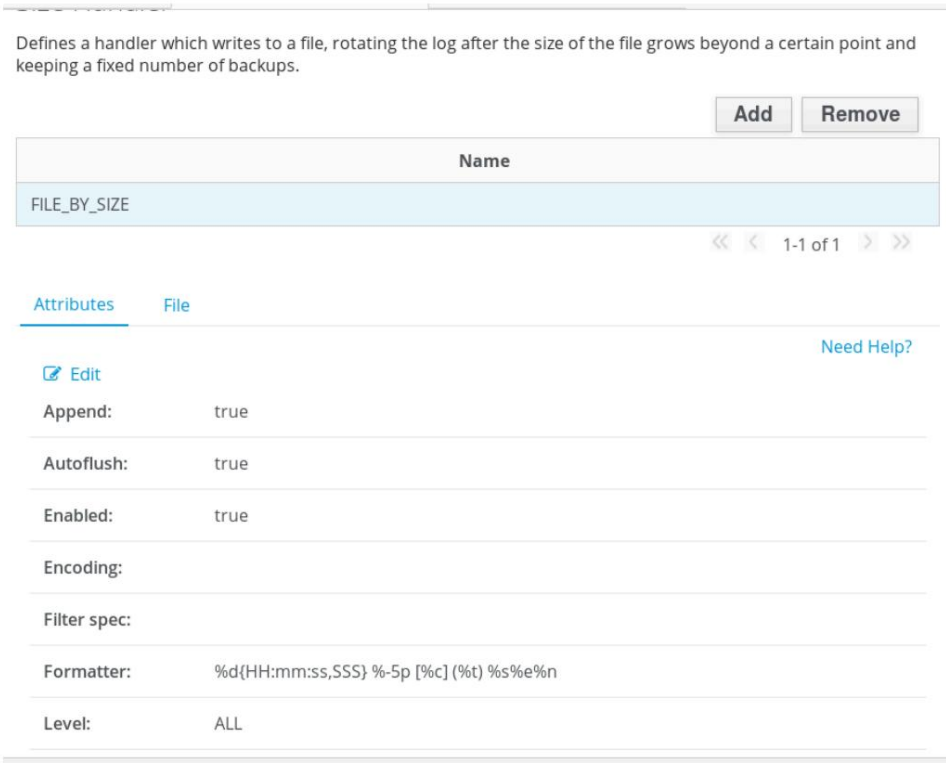


Figure 7.7: The new FILE_BY_SIZE handler

10. Edit the attributes of the FILE_BY_SIZE handler by clicking the Edit link under the Attributes tab.

Change the Level attribute to INFO in the dropdown field next to the Level label, and change the value of the Rotation Size field to 1m (1MB). Finally, click Save to modify the handler attributes.

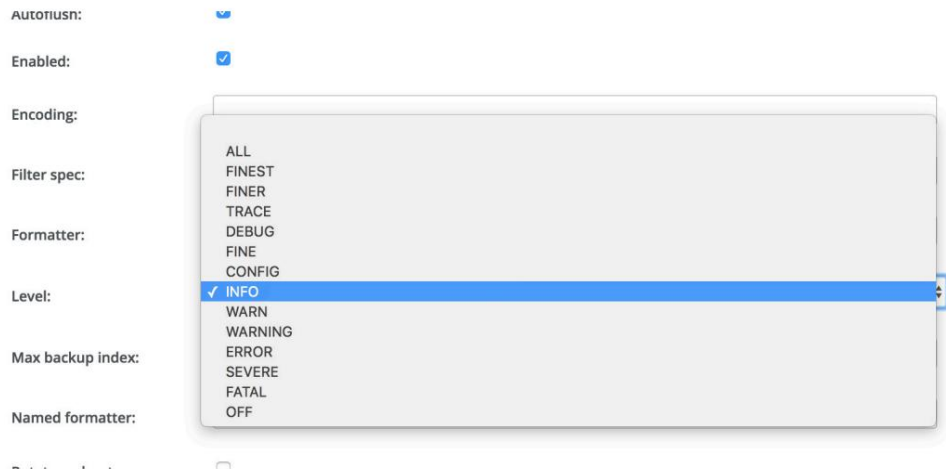


Figure 7.8: Changing the log level and handle rotation size

11. Abra el archivo `/home/student/JB248/labs/standalone/configuration/standalone.xml` and verify that the new handle for rotating files based on size called `FILE_BY_SIZE` is visible:

```
<size-rotating-file-handler name="FILE_BY_SIZE">
  <level name="INFO"/> <file
    relative-to="jboss.server.log.dir" path="production-server.log"/> <rotate-size value="1m"/> </size-rotating-file-handler>
```

We'll use this `FILE_BY_SIZE` handler in the next demo to set up asynchronous logging.

12. Verify that a file named `production-server.log` has been created in the `/home/student/JB248/labs/standalone/log` folder. The file rotates when its size is greater than the `rotate-size` attribute. The rotated file will be preserved and the index will be appended to the name which increases the indexes of previously rotated files by 1 until the `max-backup-index` is reached. Once the `max-backup-index` is reached, the indexed files will be overwritten. The default value of `max-backup-index` is 1.

13. Stop EAP by pressing `Ctrl+C` in the terminal window that is running EAP.

This concludes the demo.

asynchronous registration

The `async-handler` provides a simple way to perform *asynchronous registration*. The `async` handler does not actually write log events to a file or to the console (or any other resource). Instead, the `async-handler` sits in front of an existing handler and adds the `async` behavior. Use the `<subhandler>` element to define which handler the asynchronous behavior is added to. Here is a sample definition of this handler:

```
<async-handler name="ASYNC">
  <level name="INFO"/>
  <queue-length value="1024"/>
  <overflow-action value="BLOCK"/>
  <subhandlers>
    <handler name="FILE"/>
  </subhandlers> </
async-handler>
```

- The name of this handler is `ASYNC` and the logging level is `INFO`.
- The maximum number of log messages that can remain in the queue is `1024`.
- The possible values of `overflow-action` are `BLOCK` and `DISCARD`. If the log event queue is full when trying to add a new log event, the `BLOCK` value indicates that the current thread will block and wait for space in the queue, and the `DISCARD` value indicates that the log event will simply will be discarded.

Chapter 7. Configuring the registry subsystem

- The `<subhandlers>` section can contain multiple handlers. In this case, the asynchronous behavior is added to the default FILE handler.

Adding asynchronous behavior to the registry can improve performance on high-volume servers. If a request is made to the server that generates a registration event, a response can be sent to the client without the client having to wait for the registration event to be written to the handler.

The instructor will show how to add an asynchronous record to the FILE_BY_SIZE handler defined in the previous section:

Demo: Configure asynchronous logging

Review the video to perform the steps later. Play it again as many times as you deem necessary.

1. Open a terminal window from the workstation virtual machine (Applications > Favorites > Terminal) and run the following command to create a lab directory and verify that EAP is installed and is not currently running:

```
[student@workstation ~]$ demo logging-async setup
```

2. Users often want to be able to run multiple standalone instances without needing to install EAP multiple times on the same host. The previous demo in this chapter created an alternate directory to customize the EAP instance and leave the original installation intact.

Verify that you can see the following three folders inside the `/home/student/JB248/labs/standalone` folder:

- configuration
- deployments
- lib

3. The `jboss.server.base.dir` variable allows users to submit a directory path to use as an alternate base directory for server content. Run the following command to start the EAP server using the `standalone.sh` script in the original EAP installation, while using the new EAP configuration files:

```
[student@workstation standalone]$ cd /opt/jboss-eap-7.0/bin [student@workstation
bin]$ ./standalone.sh \-Djboss.server.base.dir=/home/
student/JB248/labs/standalone/
```

The server should start without issue with the following message:

```
17:03:30,497 INFO [org.jboss.as] (Controller Boot Thread) WFLYSRV0025: JBoss EAP 7.0.0.GA (WildFly Core
2.1.2.Final-redhat-1) started in 3313ms -Started 261 of 509 services (332 services are lazy, passive or on-demand)
```

4. In this demo we will use the JBoss EAP CLI to configure the async handler. Run the following command to log in to the CLI:


```
[student@workstation standalone]$ cd /opt/jboss-eap-7.0/bin [student@workstation
bin]$ ./jboss-cli.sh --connect
```

5. **The async handler does not write log events to a file or to the console (or any other resource). Instead, it sits in front of an existing handler and adds the async behavior. In this demo, we'll add asynchronous behavior to the FILE_BY_SIZE handler that we created in the previous demo. First, verify that the FILE_BY_SIZE handler has been defined correctly using the EAP CLI:**

```
[standalone@localhost:9990 /] /subsystem=logging\
  /size-rotating-file-handler=FILE_BY_SIZE:read-resource
{
  "outcome" => "success", "result"
  => { "append" =>
    true, "autoflush" =>
    true, "enabled" => true,
    "encoding" => undefined,
    "file" => {
      "relative-to" => "jboss.server.log.dir", "path" =>
      "production-server.log"
    },
    "filter" => undefined, "filter-
    spec" => undefined, "formatter" =>
    "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%e%n", "level" => "INFO", "max-
    backup-index" => 1,
    "name" => "FILE_BY_SIZE",
    "named-formatter" => undefined,
    "rotate-on-boot" => false, "rotate-size" =>
    "1m", "suffix" => undefined
  }
}
```

6. **The asynchronous handler has several configuration attributes:**

- **level:** the default logging level this handler will use (ALL, INFO, DEBUG, ERROR, WARN etc.)
- **queue-length** – The maximum number of log messages to keep in memory while files are flushed to disk asynchronously.
- **overflow-action:** the action the handler should take when a queue is full (DISCARD ignores log messages, BLOCK blocks the thread writing the log message until there is free space on the queue).
- **subhandler:** an existing handler to which a behavior should be added asynchronous.

Create a new async handler named ASYNC with a level set to INFO, a queue-length value of 1024, and overflow-action set to BLOCK, and assign the FILE_BY_SIZE handler to it as a subhandler:

Chapter 7. Configuring the registry subsystem

```
[standalone@localhost:9990 /] /subsystem=logging/async-handler=ASYNC:add\ (level=INFO,queue-length=1024,overflow-action=BLOCK,subhandlers=[FILE_BY_SIZE])
{"outcome" => "success"}
```

7. Add the ASYNC handler to the ROOT record:

```
[standalone@localhost:9990 /] /subsystem=logging/root-logger=ROOT:\ add-
handler(name=ASYNC)
{"outcome" => "success"}
```

8. Abra el archivo /home/student/JB248/labs/standalone/configuration/standalone.xml and verify that an async-handler named ASYNC is visible:

```
<async-handler name="ASYNC">
  <level name="INFO"/>
  <queue-length value="1024"/>
  <overflow-action value="block"/>
  <subhandlers>
    <handler name="FILE_BY_SIZE"/>
  </subhandlers>
</async-handler>
```

9. You can also verify the new async-handler using the EAP CLI:

```
[standalone@localhost:9990 /] /subsystem=logging/async-handler=ASYNC:read-resource {
  "outcome" => "success", "result"
  => { "enabled" =>
    true, "filter" => undefined,
    "filter-spec" => undefined,
    "level" => "INFO", "name" =>
    "ASYNC", "overflow-
    action" => "BLOCK",
    "queue-length" => 1024, "subhandlers"
    => ["FILE_BY_SIZE"]
  }
}
```

10. Open a new terminal window on the workstation virtual machine and note the /home/student/JB248/labs/standalone/log/production-server.log file:

```
[student@workstation ~]$ tail -f \ /home/
student/JB248/labs/standalone/log/production-server.log
```

11. Stop EAP by pressing Ctrl+C in the terminal window that is running EAP.

12. Notice that when you shut down EAP, the log messages are logged to the /home/student/JB248/labs/standalone/log/production-server.log file.

13. Restart the standalone instance of EAP and verify that the log messages are seen in the /home/student/JB248/labs/standalone/log/production server.log file.

Although you have not referenced the `FILE_BY_SIZE` handler directly, the `ASYNC` handler adds asynchronous behavior to the `FILE_BY_SIZE` handler and sends log messages to it, whereupon these log messages are logged to the `production-server.log` file via the `FILE_BY_SIZE` handler.

14. Stop EAP by pressing `Ctrl+C` in the terminal window that is running EAP.

This concludes the demo.

syslog handler

The `syslog-handler` offers a simple way to send events to a remote log server. This allows multiple applications to send their log messages to a central syslog server, where they can be simultaneously stored, analyzed, and archived to simplify backups. Here is a sample definition of this handler:

```
<syslog-handler name="SYSLOG">
  <level name="INFO"/>
  <server-address value="172.25.2.9"/> <hostname
value="workstation.lab.example.com"/> <port value="514"/>
  <app-name
value="MY_APP"/> </syslog-
handler>
```

- The name of this handler is `SYSLOG` and the log level is `INFO`.
- This handler sends log messages to the remote log server located at `172.25.2.9` and listens on TCP port `514`.
- This handler formats the log message according to the RFC5424 specification. (syslog protocol). Any syslog-compliant remote server can accept these log messages.
- The name of the host from which messages are being sent is set to `workstation.lab.example.com` to uniquely identify this host, since multiple hosts can send syslog messages to the remote server. Log messages on the remote server can be parsed and filtered based on this attribute.

Syslog handlers can be defined using the EAP management console in the Configuration > Subsystems > Logging section. Click Log, the Handler tab, and then the Syslog link to view, edit, or add a syslog-handler:

Chapter 7. Configuring the registry subsystem

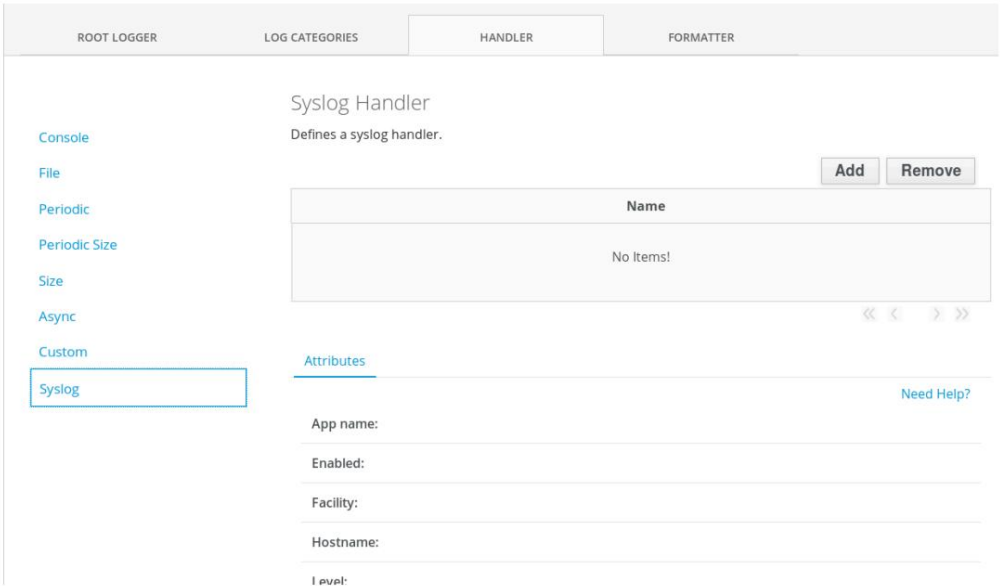


Figure 7.9: Syslog Manager