Spring Boot Actuator Web API Documentation

Andy Wilkinson

Table of Contents

This API documentation describes Spring Boot Actuators web endpoints.

Chapter 1. Overview

Before you proceed, you should read the following topics:

- URLs
- Timestamps

1.1. URLs

By default, all web endpoints are available beneath the path `/actuator` with URLs of the form `/actuator/{id}`. The `/actuator` base path can be configured by using the `management.endpoints.web.base-path` property, as shown in the following example:

```
management.endpoints.web.base-path=/manage
```

The preceding `application.properties` example changes the form of the endpoint URLs from `/actuator/{id}` to `/manage/{id}`. For example, the URL `info` endpoint would become `/manage/info`.

1.2. Timestamps

All timestamps that are consumed by the endpoints, either as query parameters or in the request body, must be formatted as an offset date and time as specified in ISO 8601.

Chapter 2. Audit Events (auditevents)

The auditevents endpoint provides information about the application's audit events.

2.1. Retrieving Audit Events

To retrieve the audit events, make a GET request to /actuator/auditevents, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/auditevents?principal=alice&after=2019-01-
12T02%3A28%3A43.341Z&type=logout' -i -X GET
```

The preceding example retrieves logout events for the principal, alice, that occurred after 09:37 on 7 November 2017 in the UTC timezone. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 121

{
  "events" : [ {
    "timestamp" : "2019-01-12T02:28:43.343Z",
    "principal" : "alice",
    "type" : "logout"
  } ]
}
```

2.1.1. Query Parameters

The endpoint uses query parameters to limit the events that it returns. The following table shows the supported query parameters:

Parameter	Description
after	Restricts the events to those that occurred after the given time. Optional.
principal	Restricts the events to those with the given principal. Optional.
type	Restricts the events to those with the given type. Optional.

2.1.2. Response Structure

The response contains details of all of the audit events that matched the query. The following table describes the structure of the response:

Path	Type	Description
`events`	`Array`	An array of audit events.
`events.[].timestamp`	`String`	The timestamp of when the event occurred.
`events.[].principal`	`String`	The principal that triggered the event.
`events.[].type`	`String`	The type of the event.

Chapter 3. Beans (beans)

The beans endpoint provides information about the application's beans.

3.1. Retrieving the Beans

To retrieve the beans, make a GET request to /actuator/beans, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/beans' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 1128

{
  "contexts" : {
    "application" : {
      "beans" : {
        "defaultServletHandlerMapping" : {
          "aliases" : [ ],
          "scope" : "singleton",
          "type" : "org.springframework.web.servlet.HandlerMapping",
          "resource" : "class path resource
[org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoConfiguration$EnableWebM
vcConfiguration.class]",
          "dependencies" : [ ]
        },

"org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration"
: {
          "aliases" : [ ],
          "scope" : "singleton",
          "type" :
"org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration$$
EnhancerBySpringCGLIB$$ca382217",
          "dependencies" : [ ]
        },

"org.springframework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration
" : {
          "aliases" : [ ],
          "scope" : "singleton",
          "type" :
"org.springframework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration
$$EnhancerBySpringCGLIB$$b7e4049b",
          "dependencies" : [ ]
        }
      }
    }
  }
}
```

3.1.1. Response Structure

The response contains details of the application's beans. The following table describes the structure of the response:

Path	Type	Description
`contexts`	`Object`	Application contexts keyed by id.
`contexts.*.parentId`	`String`	Id of the parent application context, if any.
`contexts.*.beans`	`Object`	Beans in the application context keyed by name.
`contexts.*.beans.*.aliases`	`Array`	Names of any aliases.
`contexts.*.beans.*.scope`	`String`	Scope of the bean.
`contexts.*.beans.*.type`	`String`	Fully qualified type of the bean.
`contexts.*.beans.*.resource`	`String`	Resource in which the bean was defined, if any.
`contexts.*.beans.*.dependencies`	`Array`	Names of any dependencies.

Chapter 4. Caches (caches)

The caches endpoint provides access to the application's caches.

4.1. Retrieving All Caches

To retrieve the application's caches, make a GET request to /actuator/caches, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/caches' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 435

{
  "cacheManagers" : {
    "anotherCacheManager" : {
      "caches" : {
        "countries" : {
          "target" : "java.util.concurrent.ConcurrentHashMap"
        }
      }
    },
    "cacheManager" : {
      "caches" : {
        "cities" : {
          "target" : "java.util.concurrent.ConcurrentHashMap"
        },
        "countries" : {
          "target" : "java.util.concurrent.ConcurrentHashMap"
        }
      }
    }
  }
}
```

4.1.1. Response Structure

The response contains details of the application's caches. The following table describes the structure of the response:

Path	Type	Description
cacheManagers	Object	Cache managers keyed by id.

Path	Type	Description
`cacheManagers.*.caches`	`Object`	Caches in the application context keyed by name.
`cacheManagers.*.caches.*.target`	`String`	Fully qualified name of the native cache.

4.2. Retrieving Caches by Name

To retrieve a cache by name, make a `GET` request to `/actuator/caches/{name}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/caches/cities' -i -X GET
```

The preceding example retrieves information about the cache named `cities`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 113

{
  "target" : "java.util.concurrent.ConcurrentHashMap",
  "name" : "cities",
  "cacheManager" : "cacheManager"
}
```

4.2.1. Query Parameters

If the requested name is specific enough to identify a single cache, no extra parameter is required. Otherwise, the `cacheManager` must be specified. The following table shows the supported query parameters:

Parameter	Description
`cacheManager`	Name of the cacheManager to qualify the cache. May be omitted if the cache name is unique.

4.2.2. Response Structure

The response contains details of the requested cache. The following table describes the structure of the response:

Path	Type	Description
`name`	`String`	Cache name.
`cacheManager`	`String`	Cache manager name.

Path	Type	Description
`target`	`String`	Fully qualified name of the native cache.

4.3. Evict All Caches

To clear all available caches, make a `DELETE` request to `/actuator/caches` as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/caches' -i -X DELETE
```

4.4. Evict a Cache by Name

To evict a particular cache, make a `DELETE` request to `/actuator/caches/{name}` as shown in the following curl-based example:

```
$ curl
'http://localhost:8080/actuator/caches/countries?cacheManager=anotherCacheManager' -i
-X DELETE
```

As there are two caches named `countries`, the `cacheManager` has to be provided to specify which `Cache` should be cleared.

4.4.1. Request Structure

If the requested name is specific enough to identify a single cache, no extra parameter is required. Otherwise, the `cacheManager` must be specified. The following table shows the supported query parameters:

Parameter	Description
`cacheManager`	Name of the cacheManager to qualify the cache. May be omitted if the cache name is unique.

Chapter 5. Conditions Evaluation Report (`conditions`)

The `conditions` endpoint provides information about the evaluation of conditions on configuration and auto-configuration classes.

5.1. Retrieving the Report

To retrieve the report, make a `GET` request to `/actuator/conditions`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/conditions' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 3259

{
  "contexts" : {
    "application" : {
      "positiveMatches" : {
        "EndpointAutoConfiguration#endpointOperationParameterMapper" : [ {
          "condition" : "OnBeanCondition",
          "message" : "@ConditionalOnMissingBean (types:
org.springframework.boot.actuate.endpoint.invoke.ParameterValueMapper; SearchStrategy:
all) did not find any beans"
        } ],
        "EndpointAutoConfiguration#endpointCachingOperationInvokerAdvisor" : [ {
          "condition" : "OnBeanCondition",
          "message" : "@ConditionalOnMissingBean (types:
org.springframework.boot.actuate.endpoint.invoker.cache.CachingOperationInvokerAdvisor
; SearchStrategy: all) did not find any beans"
        } ],
        "WebEndpointAutoConfiguration" : [ {
          "condition" : "OnWebApplicationCondition",
          "message" : "@ConditionalOnWebApplication (required) found 'session' scope"
        } ]
      },
      "negativeMatches" : {
        "WebFluxEndpointManagementContextConfiguration" : {
          "notMatched" : [ {
            "condition" : "OnWebApplicationCondition",
            "message" : "not a reactive web application"
          } ],
          "matched" : [ {
```

```
            "condition" : "OnClassCondition",
            "message" : "@ConditionalOnClass found required classes
'org.springframework.web.reactive.DispatcherHandler',
'org.springframework.http.server.reactive.HttpHandler'"
          } ]
        },
        "GsonHttpMessageConvertersConfiguration.GsonHttpMessageConverterConfiguration"
: {
          "notMatched" : [ {
            "condition" :
"GsonHttpMessageConvertersConfiguration.PreferGsonOrJacksonAndJsonbUnavailableConditio
n",
            "message" : "AnyNestedCondition 0 matched 2 did not; NestedCondition on
GsonHttpMessageConvertersConfiguration.PreferGsonOrJacksonAndJsonbUnavailableCondition
.JacksonJsonbUnavailable NoneNestedConditions 1 matched 1 did not; NestedCondition on
GsonHttpMessageConvertersConfiguration.JacksonAndJsonbUnavailableCondition.JsonbPrefer
red @ConditionalOnProperty (spring.http.converters.preferred-json-mapper=jsonb) did
not find property 'spring.http.converters.preferred-json-mapper'; NestedCondition on
GsonHttpMessageConvertersConfiguration.JacksonAndJsonbUnavailableCondition.JacksonAvai
lable @ConditionalOnBean (types:
org.springframework.http.converter.json.MappingJackson2HttpMessageConverter;
SearchStrategy: all) found bean 'mappingJackson2HttpMessageConverter'; NestedCondition
on
GsonHttpMessageConvertersConfiguration.PreferGsonOrJacksonAndJsonbUnavailableCondition
.GsonPreferred @ConditionalOnProperty (spring.http.converters.preferred-json-
mapper=gson) did not find property 'spring.http.converters.preferred-json-mapper'"
          } ],
          "matched" : [ ]
        },
        "JsonbHttpMessageConvertersConfiguration" : {
          "notMatched" : [ {
            "condition" : "OnClassCondition",
            "message" : "@ConditionalOnClass did not find required class
'javax.json.bind.Jsonb'"
          } ],
          "matched" : [ ]
        }
      },
      "unconditionalClasses" : [
"org.springframework.boot.autoconfigure.context.PropertyPlaceholderAutoConfiguration",
"org.springframework.boot.actuate.autoconfigure.endpoint.EndpointAutoConfiguration" ]
    }
  }
}
```


5.1.1. Response Structure

The response contains details of the application's condition evaluation. The following table describes the structure of the response:

Path	Type	Description
contexts	Object	Application contexts keyed by id.
contexts.*.positiveMatches	Object	Classes and methods with conditions that were matched.
contexts.*.positiveMatches.*.[].condition	String	Name of the condition.
contexts.*.positiveMatches.*.[].message	String	Details of why the condition was matched.
contexts.*.negativeMatches	Object	Classes and methods with conditions that were not matched.
contexts.*.negativeMatches.*.notMatched	Array	Conditions that were matched.
contexts.*.negativeMatches.*.notMatched.[].condition	String	Name of the condition.
contexts.*.negativeMatches.*.notMatched.[].message	String	Details of why the condition was not matched.
contexts.*.negativeMatches.*.matched	Array	Conditions that were matched.
contexts.*.negativeMatches.*.matched.[].condition	String	Name of the condition.
contexts.*.negativeMatches.*.matched.[].message	String	Details of why the condition was matched.
contexts.*.unconditionalClasses	Array	Names of unconditional auto-configuration classes if any.
contexts.*.parentId	String	Id of the parent application context, if any.

Chapter 6. Configuration Properties (configprops)

The configprops endpoint provides information about the application's @ConfigurationProperties beans.

6.1. Retrieving the @ConfigurationProperties Bean

To retrieve the @ConfigurationProperties beans, make a GET request to /actuator/configprops, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/configprops' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 1806

{
  "contexts" : {
    "application" : {
      "beans" : {
        "management.endpoints.web.cors-
org.springframework.boot.actuate.autoconfigure.endpoint.web.CorsEndpointProperties" :
{
          "prefix" : "management.endpoints.web.cors",
          "properties" : {
            "allowedHeaders" : [ ],
            "allowedMethods" : [ ],
            "allowedOrigins" : [ ],
            "maxAge" : {
              "units" : [ "SECONDS", "NANOS" ]
            },
            "exposedHeaders" : [ ]
          }
        },
        "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointProperties" : {
          "prefix" : "management.endpoints.web",
          "properties" : {
            "pathMapping" : { },
            "exposure" : {
              "include" : [ "*" ],
              "exclude" : [ ]
            },
            "basePath" : "/actuator"
```

```
          }
        },
        "spring.resources-
org.springframework.boot.autoconfigure.web.ResourceProperties" : {
          "prefix" : "spring.resources",
          "properties" : {
            "addMappings" : true,
            "chain" : {
              "cache" : true,
              "htmlApplicationCache" : false,
              "compressed" : false,
              "strategy" : {
                "fixed" : {
                  "enabled" : false,
                  "paths" : [ "/**" ]
                },
                "content" : {
                  "enabled" : false,
                  "paths" : [ "/**" ]
                }
              }
            },
            "cache" : {
              "cachecontrol" : { }
            },
            "staticLocations" : [ "classpath:/META-INF/resources/",
"classpath:/resources/", "classpath:/static/", "classpath:/public/" ]
          }
        }
      }
    }
  }
}
```

6.1.1. Response Structure

The response contains details of the application's `@ConfigurationProperties` beans. The following table describes the structure of the response:

Path	Type	Description
`contexts`	`Object`	Application contexts keyed by id.
`contexts.*.beans.*`	`Object`	`@ConfigurationProperties` beans keyed by bean name.
`contexts.*.beans.*.prefix`	`String`	Prefix applied to the names of the bean's properties.
`contexts.*.beans.*.properties`	`Object`	Properties of the bean as name-value pairs.
`contexts.*.parentId`	`String`	Id of the parent application context, if any.

Chapter 7. Environment (env)

The `env` endpoint provides information about the application's `Environment`.

7.1. Retrieving the Entire Environment

To retrieve the entire environment, make a `GET` request to `/actuator/env`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/env' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 794

{
  "activeProfiles" : [ ],
  "propertySources" : [ {
    "name" : "systemProperties",
    "properties" : {
      "java.runtime.name" : {
        "value" : "OpenJDK Runtime Environment"
      },
      "java.vm.version" : {
        "value" : "25.192-b12"
      },
      "java.vm.vendor" : {
        "value" : "Oracle Corporation"
      }
    }
  }, {
    "name" : "systemEnvironment",
    "properties" : {
      "JAVA_HOME" : {
        "value" : "/opt/openjdk",
        "origin" : "System Environment Property \"JAVA_HOME\""
      }
    }
  }, {
    "name" : "applicationConfig: [classpath:/application.properties]",
    "properties" : {
      "com.example.cache.max-size" : {
        "value" : "1000",
        "origin" : "class path resource [application.properties]:1:29"
      }
    }
  } ]
}
```

7.1.1. Response Structure

The response contains details of the application's `Environment`. The following table describes the structure of the response:

Path	Type	Description
`activeProfiles`	`Array`	Names of the active profiles, if any.
`propertySources`	`Array`	Property sources in order of precedence.
`propertySources.[].name`	`String`	Name of the property source.

Path	Type	Description
`propertySources.[].properties`	`Object`	Properties in the property source keyed by property name.
`propertySources.[].properties.*.value`	`String`	Value of the property.
`propertySources.[].properties.*.origin`	`String`	Origin of the property, if any.

7.2. Retrieving a Single Property

To retrieve a single property, make a `GET` request to `/actuator/env/{property.name}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/env/com.example.cache.max-size' -i -X GET
```

The preceding example retrieves information about the property named `com.example.cache.max-size`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 445
Content-Disposition: inline;filename=f.txt

{
  "property" : {
    "source" : "applicationConfig: [classpath:/application.properties]",
    "value" : "1000"
  },
  "activeProfiles" : [ ],
  "propertySources" : [ {
    "name" : "systemProperties"
  }, {
    "name" : "systemEnvironment"
  }, {
    "name" : "applicationConfig: [classpath:/application.properties]",
    "property" : {
      "value" : "1000",
      "origin" : "class path resource [application.properties]:1:29"
    }
  } ]
}
```

7.2.1. Response Structure

The response contains details of the requested property. The following table describes the structure of the response:

Path	Type	Description
`property`	`Object`	Property from the environment, if found.
`property.source`	`String`	Name of the source of the property.
`property.value`	`String`	Value of the property.
`activeProfiles`	`Array`	Names of the active profiles, if any.
`propertySources`	`Array`	Property sources in order of precedence.
`propertySources.[].name`	`String`	Name of the property source.
`propertySources.[].property`	`Object`	Property in the property source, if any.
`propertySources.[].property.value`	`Varies`	Value of the property.
`propertySources.[].property.origin`	`String`	Origin of the property, if any.

Chapter 8. Flyway (`flyway`)

The `flyway` endpoint provides information about database migrations performed by Flyway.

8.1. Retrieving the Migrations

To retrieve the migrations, make a `GET` request to `/actuator/flyway`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/flyway' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 515

{
  "contexts" : {
    "application" : {
      "flywayBeans" : {
        "flyway" : {
          "migrations" : [ {
            "type" : "SQL",
            "checksum" : -156244537,
            "version" : "1",
            "description" : "init",
            "script" : "V1__init.sql",
            "state" : "SUCCESS",
            "installedBy" : "SA",
            "installedOn" : "2019-01-12T02:28:49.366Z",
            "installedRank" : 1,
            "executionTime" : 1
          } ]
        }
      }
    }
  }
}
```

8.1.1. Response Structure

The response contains details of the application's Flyway migrations. The following table describes the structure of the response:

Path	Type	Description
`contexts`	`Object`	Application contexts keyed by id
`contexts.*.flywayBeans.*.migrations`	`Array`	Migrations performed by the Flyway instance, keyed by Flyway bean name.
`contexts.*.flywayBeans.*.migrations.[].checksum`	`Number`	Checksum of the migration, if any.
`contexts.*.flywayBeans.*.migrations.[].description`	`String`	Description of the migration, if any.
`contexts.*.flywayBeans.*.migrations.[].executionTime`	`Number`	Execution time in milliseconds of an applied migration.
`contexts.*.flywayBeans.*.migrations.[].installedBy`	`String`	User that installed the applied migration, if any.
`contexts.*.flywayBeans.*.migrations.[].installedOn`	`String`	Timestamp of when the applied migration was installed, if any.
`contexts.*.flywayBeans.*.migrations.[].installedRank`	`Number`	Rank of the applied migration, if any. Later migrations have higher ranks.
`contexts.*.flywayBeans.*.migrations.[].script`	`String`	Name of the script used to execute the migration, if any.
`contexts.*.flywayBeans.*.migrations.[].state`	`String`	State of the migration. (`PENDING`, `ABOVE_TARGET`, `BELOW_BASELINE`, `BASELINE`, `IGNORED`, `MISSING_SUCCESS`, `MISSING_FAILED`, `SUCCESS`, `UNDONE`, `AVAILABLE`, `FAILED`, `OUT_OF_ORDER`, `FUTURE_SUCCESS`, `FUTURE_FAILED`, `OUTDATED`, `SUPERSEDED`)
`contexts.*.flywayBeans.*.migrations.[].type`	`String`	Type of the migration. (`SCHEMA`, `BASELINE`, `SQL`, `UNDO_SQL`, `JDBC`, `UNDO_JDBC`, `SPRING_JDBC`, `UNDO_SPRING_JDBC`, `CUSTOM`, `UNDO_CUSTOM`)
`contexts.*.flywayBeans.*.migrations.[].version`	`String`	Version of the database after applying the migration, if any.
`contexts.*.parentId`	`String`	Id of the parent application context, if any.

Chapter 9. Health (`health`)

The `health` endpoint provides detailed information about the health of the application.

9.1. Retrieving the Health of the application

To retrieve the health of the application, make a `GET` request to `/actuator/health`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/health' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 670

{
  "status" : "UP",
  "details" : {
    "diskSpace" : {
      "status" : "UP",
      "details" : {
        "total" : 162337054720,
        "free" : 65662001152,
        "threshold" : 10485760
      }
    },
    "db" : {
      "status" : "UP",
      "details" : {
        "database" : "HSQL Database Engine",
        "hello" : 1
      }
    },
    "broker" : {
      "status" : "UP",
      "details" : {
        "us1" : {
          "status" : "UP",
          "details" : {
            "version" : "1.0.2"
          }
        },
        "us2" : {
          "status" : "UP",
          "details" : {
            "version" : "1.0.4"
          }
        }
      }
    }
  }
}
```

9.1.1. Response Structure

The response contains details of the health of the application. The following table describes the structure of the response:

Path	Type	Description
`status`	`String`	Overall status of the application.
`details`	`Object`	Details of the health of the application. Presence is controlled by `management.endpoint.health.show-details`).
`details.*.status`	`String`	Status of a specific part of the application.
`details.*.details`	`Object`	Details of the health of a specific part of the application.

9.2. Retrieving the Health of a component

To retrieve the health of a particular component of the application, make a `GET` request to `/actuator/health/{component}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/health/db' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 99

{
  "status" : "UP",
  "details" : {
    "database" : "HSQL Database Engine",
    "hello" : 1
  }
}
```

9.2.1. Response Structure

The response contains details of the health of a particular component of the application. The following table describes the structure of the response:

Path	Type	Description
`status`	`String`	Status of a specific part of the application
`details`	`Object`	Details of the health of a specific part of the application.

9.3. Retrieving the Health of a component instance

If a particular component consists of multiple instances (as the `broker` indicator in the example above), the health of a particular instance of that component can be retrieved by issuing a `GET` request to `/actuator/health/{component}/{instance}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/health/broker/us1' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 66

{
  "status" : "UP",
  "details" : {
    "version" : "1.0.2"
  }
}
```

9.3.1. Response Structure

The response contains details of the health of an instance of a particular component of the application. The following table describes the structure of the response:

Path	Type	Description
`status`	`String`	Status of a specific part of the application
`details`	`Object`	Details of the health of a specific part of the application.

Chapter 10. Heap Dump (heapdump)

The heapdump endpoint provides a heap dump from the application's JVM.

10.1. Retrieving the Heap Dump

To retrieve the heap dump, make a GET request to /actuator/heapdump. The response is binary data in HPROF format and can be large. Typically, you should save the response to disk for subsequent analysis. When using curl, this can be achieved by using the -O option, as shown in the following example:

```
$ curl 'http://localhost:8080/actuator/heapdump' -O
```

The preceding example results in a file named heapdump being written to the current working directory.

Chapter 11. HTTP Trace (`httptrace`)

The `httptrace` endpoint provides information about HTTP request-response exchanges.

11.1. Retrieving the Traces

To retrieve the traces, make a `GET` request to `/actuator/httptrace`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/httptrace' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 503

{
  "traces" : [ {
    "timestamp" : "2019-01-12T02:28:51.894Z",
    "principal" : {
      "name" : "alice"
    },
    "session" : {
      "id" : "bf9598e3-b149-45f9-a85b-82a5d6d54f12"
    },
    "request" : {
      "method" : "GET",
      "uri" : "https://api.example.com",
      "headers" : {
        "Accept" : [ "application/json" ]
      }
    },
    "response" : {
      "status" : 200,
      "headers" : {
        "Content-Type" : [ "application/json" ]
      }
    },
    "timeTaken" : 1
  } ]
}
```

11.1.1. Response Structure

The response contains details of the traced HTTP request-response exchanges. The following table describes the structure of the response:

Path	Type	Description
traces	Array	An array of traced HTTP request-response exchanges.
traces.[].timestamp	String	Timestamp of when the traced exchange occurred.
traces.[].principal	Object	Principal of the exchange, if any.
traces.[].principal.name	String	Name of the principal.
traces.[].request.method	String	HTTP method of the request.
traces.[].request.remoteAddress	String	Remote address from which the request was received, if known.
traces.[].request.uri	String	URI of the request.
traces.[].request.headers	Object	Headers of the request, keyed by header name.
traces.[].request.headers.*.[]	Array	Values of the header
traces.[].response.status	Number	Status of the response
traces.[].response.headers	Object	Headers of the response, keyed by header name.
traces.[].response.headers.*.[]	Array	Values of the header
traces.[].session	Object	Session associated with the exchange, if any.
traces.[].session.id	String	ID of the session.
traces.[].timeTaken	Number	Time, in milliseconds, taken to handle the exchange.

Chapter 12. Info (`info`)

The `info` endpoint provides general information about the application.

12.1. Retrieving the Info

To retrieve the information about the application, make a `GET` request to `/actuator/info`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/info' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 235

{
  "git" : {
    "commit" : {
      "time" : "+51000-09-21T09:09:07Z",
      "id" : "df027cf"
    },
    "branch" : "master"
  },
  "build" : {
    "version" : "1.0.3",
    "artifact" : "application",
    "group" : "com.example"
  }
}
```

12.1.1. Response Structure

The response contains general information about the application. Each section of the response is contributed by an `InfoContributor`. Spring Boot provides `build` and `git` contributions.

`build` Response Structure

The following table describe the structure of the `build` section of the response:

Path	Type	Description
`artifact`	`String`	Artifact ID of the application, if any.
`group`	`String`	Group ID of the application, if any.
`name`	`String`	Name of the application, if any.

Path	Type	Description
`version`	`String`	Version of the application, if any.
`time`	`Varies`	Timestamp of when the application was built, if any.

`git` Response Structure

The following table describes the structure of the `git` section of the response:

Path	Type	Description
`branch`	`String`	Name of the Git branch, if any.
`commit`	`Object`	Details of the Git commit, if any.
`commit.time`	`Varies`	Timestamp of the commit, if any.
`commit.id`	`String`	ID of the commit, if any.

Chapter 13. Spring Integration graph (`integrationgraph`)

The `integrationgraph` endpoint exposes a graph containing all Spring Integration components.

13.1. Retrieving the Spring Integration graph

To retrieve the information about the application, make a `GET` request to `/actuator/integrationgraph`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/integrationgraph' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 667

{
  "contentDescriptor" : {
    "providerVersion" : "5.1.2.RELEASE",
    "providerFormatVersion" : 1.0,
    "provider" : "spring-integration"
  },
  "nodes" : [ {
    "nodeId" : 1,
    "name" : "nullChannel",
    "componentType" : "null-channel",
    "properties" : { }
  }, {
    "nodeId" : 2,
    "name" : "errorChannel",
    "componentType" : "publish-subscribe-channel",
    "properties" : { }
  }, {
    "nodeId" : 3,
    "name" : "_org.springframework.integration.errorLogger",
    "componentType" : "logging-channel-adapter",
    "properties" : { },
    "input" : "errorChannel"
  } ],
  "links" : [ {
    "from" : 2,
    "to" : 3,
    "type" : "input"
  } ]
}
```

13.1.1. Response Structure

The response contains all Spring Integration components used within the application, as well as the links between them. More information about the structure can be found in the reference documentation.

13.2. Rebuilding the Spring Integration graph

To rebuild the exposed graph, make a `POST` request to `/actuator/integrationgraph`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/integrationgraph' -i -X POST
```

This will result in a `204 - No Content` response:

```
HTTP/1.1 204 No Content
```

Chapter 14. Liquibase (`liquibase`)

The `liquibase` endpoint provides information about database change sets applied by Liquibase.

14.1. Retrieving the Changes

To retrieve the changes, make a `GET` request to `/actuator/liquibase`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/liquibase' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 688

{
  "contexts" : {
    "application" : {
      "liquibaseBeans" : {
        "liquibase" : {
          "changeSets" : [ {
            "author" : "marceloverdijk",
            "changeLog" : "classpath:/db/changelog/db.changelog-master.yaml",
            "comments" : "",
            "contexts" : [ ],
            "dateExecuted" : "2019-01-12T02:28:06.658Z",
            "deploymentId" : "7260086629",
            "description" : "createTable tableName=customer",
            "execType" : "EXECUTED",
            "id" : "1",
            "labels" : [ ],
            "checksum" : "8:46debf252cce6d7b25e28ddeb9fc4bf6",
            "orderExecuted" : 1
          } ]
        }
      }
    }
  }
}
```

14.1.1. Response Structure

The response contains details of the application's Liquibase change sets. The following table describes the structure of the response:

Path	Type	Description
`contexts`	`Object`	Application contexts keyed by id
`contexts.*.liquibaseBeans.*.changeSets`	`Array`	Change sets made by the Liquibase beans, keyed by bean name.
`contexts.*.liquibaseBeans.*.changeSets[].author`	`String`	Author of the change set.
`contexts.*.liquibaseBeans.*.changeSets[].changeLog`	`String`	Change log that contains the change set.
`contexts.*.liquibaseBeans.*.changeSets[].comments`	`String`	Comments on the change set.
`contexts.*.liquibaseBeans.*.changeSets[].contexts`	`Array`	Contexts of the change set.
`contexts.*.liquibaseBeans.*.changeSets[].dateExecuted`	`String`	Timestamp of when the change set was executed.
`contexts.*.liquibaseBeans.*.changeSets[].deploymentId`	`String`	ID of the deployment that ran the change set.
`contexts.*.liquibaseBeans.*.changeSets[].description`	`String`	Description of the change set.
`contexts.*.liquibaseBeans.*.changeSets[].execType`	`String`	Execution type of the change set (`EXECUTED`, `FAILED`, `SKIPPED`, `RERAN`, `MARK_RAN`).
`contexts.*.liquibaseBeans.*.changeSets[].id`	`String`	ID of the change set.
`contexts.*.liquibaseBeans.*.changeSets[].labels`	`Array`	Labels associated with the change set.
`contexts.*.liquibaseBeans.*.changeSets[].checksum`	`String`	Checksum of the change set.
`contexts.*.liquibaseBeans.*.changeSets[].orderExecuted`	`Number`	Order of the execution of the change set.
`contexts.*.liquibaseBeans.*.changeSets[].tag`	`String`	Tag associated with the change set, if any.
`contexts.*.parentId`	`String`	Id of the parent application context, if any.

Chapter 15. Log File (`logfile`)

The `logfile` endpoint provides access to the contents of the application's log file.

15.1. Retrieving the Log File

To retrieve the log file, make a `GET` request to `/actuator/logfile`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/logfile' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Length: 4723
Content-Type: text/plain


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::

2017-08-08 17:12:30.910  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : Starting SampleWebFreeMarkerApplication on
host.local with PID 19866
2017-08-08 17:12:30.913  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : No active profile set, falling back to
default profiles: default
2017-08-08 17:12:30.952  INFO 19866 --- [           main]
ConfigServletWebServerApplicationContext : Refreshing
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicati
onContext@76b10754: startup date [Tue Aug 08 17:12:30 BST 2017]; root of context
hierarchy
2017-08-08 17:12:31.878  INFO 19866 --- [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080
(http)
2017-08-08 17:12:31.889  INFO 19866 --- [           main]
o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2017-08-08 17:12:31.890  INFO 19866 --- [           main]
org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache
Tomcat/8.5.16
2017-08-08 17:12:31.978  INFO 19866 --- [ost-startStop-1]
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded
WebApplicationContext
```

```
2017-08-08 17:12:31.978  INFO 19866 --- [ost-startStop-1]
o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization
completed in 1028 ms
2017-08-08 17:12:32.080  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.ServletRegistrationBean  : Mapping servlet: 'dispatcherServlet' to [/]
2017-08-08 17:12:32.084  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter'
to: [/*]
2017-08-08 17:12:32.084  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter'
to: [/*]
2017-08-08 17:12:32.084  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter'
to: [/*]
2017-08-08 17:12:32.084  INFO 19866 --- [ost-startStop-1]
o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to:
[/*]
2017-08-08 17:12:32.349  INFO 19866 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice:
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicati
onContext@76b10754: startup date [Tue Aug 08 17:12:30 BST 2017]; root of context
hierarchy
2017-08-08 17:12:32.420  INFO 19866 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public
org.springframework.http.ResponseEntity<java.util.Map<java.lang.String,
java.lang.Object>>
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(ja
vax.servlet.http.HttpServletRequest)
2017-08-08 17:12:32.421  INFO 19866 --- [           main]
s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}"
onto public org.springframework.web.servlet.ModelAndView
org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtm
l(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)
2017-08-08 17:12:32.444  INFO 19866 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler
of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2017-08-08 17:12:32.444  INFO 19866 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type
[class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2017-08-08 17:12:32.471  INFO 19866 --- [           main]
o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto
handler of type [class
org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2017-08-08 17:12:32.600  INFO 19866 --- [           main]
o.s.w.s.v.f.FreeMarkerConfigurer         : ClassTemplateLoader for Spring macros added
to FreeMarker configuration
2017-08-08 17:12:32.681  INFO 19866 --- [           main]
o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on
startup
2017-08-08 17:12:32.744  INFO 19866 --- [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http)
```

```
2017-08-08 17:12:32.750  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : Started SampleWebFreeMarkerApplication in
2.172 seconds (JVM running for 2.479)
```

15.2. Retrieving Part of the Log File

Retrieving part of the log file is not supported when using Jersey.

To retrieve part of the log file, make a `GET` request to `/actuator/logfile` by using the `Range` header, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/logfile' -i -X GET \
    -H 'Range: bytes=0-1023'
```

The preceding example retrieves the first 1024 bytes of the log file. The resulting response is similar to the following:

```
HTTP/1.1 206 Partial Content
Accept-Ranges: bytes
Content-Range: bytes 0-1023/4723
Content-Length: 1024
Content-Type: text/plain

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::

2017-08-08 17:12:30.910  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : Starting SampleWebFreeMarkerApplication on
host.local with PID 19866
2017-08-08 17:12:30.913  INFO 19866 --- [           main]
s.f.SampleWebFreeMarkerApplication       : No active profile set, falling back to
default profiles: default
2017-08-08 17:12:30.952  INFO 19866 --- [           main]
ConfigServletWebServerApplicationContext : Refreshing
org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicati
onContext@76b10754: startup date [Tue Aug 08 17:12:30 BST 2017]; root of context
hierarchy
2017-08-08 17:12:31.878  INFO 19866 --- [           main]
o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(
```

Chapter 16. Loggers (`loggers`)

The `loggers` endpoint provides access to the application's loggers and the configuration of their levels.

16.1. Retrieving All Loggers

To retrieve the application's loggers, make a `GET` request to `/actuator/loggers`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/loggers' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Length: 283
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8

{
  "levels" : [ "OFF", "FATAL", "ERROR", "WARN", "INFO", "DEBUG", "TRACE" ],
  "loggers" : {
    "ROOT" : {
      "configuredLevel" : "INFO",
      "effectiveLevel" : "INFO"
    },
    "com.example" : {
      "configuredLevel" : "DEBUG",
      "effectiveLevel" : "DEBUG"
    }
  }
}
```

16.1.1. Response Structure

The response contains details of the application's loggers. The following table describes the structure of the response:

Path	Type	Description
`levels`	`Array`	Levels support by the logging system.
`loggers`	`Object`	Loggers keyed by name.
`loggers.*.configuredLevel`	`String`	Configured level of the logger, if any.
`loggers.*.effectiveLevel`	`String`	Effective level of the logger.

16.2. Retrieving a Single Logger

To retrieve a single logger, make a `GET` request to `/actuator/loggers/{logger.name}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/loggers/com.example' -i -X GET
```

The preceding example retrieves information about the logger named `com.example`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 61
Content-Disposition: inline;filename=f.txt

{
  "configuredLevel" : "INFO",
  "effectiveLevel" : "INFO"
}
```

16.2.1. Response Structure

The response contains details of the requested logger. The following table describes the structure of the response:

Path	Type	Description
`configuredLevel`	`String`	Configured level of the logger, if any.
`effectiveLevel`	`String`	Effective level of the logger.

16.3. Setting a Log Level

To set the level of a logger, make a `POST` request to `/actuator/loggers/{logger.name}` with a JSON body that specifies the configured level for the logger, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/loggers/com.example' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{"configuredLevel":"debug"}'
```

The preceding example sets the `configuredLevel` of the `com.example` logger to `DEBUG`.

16.3.1. Request Structure

The request specifies the desired level of the logger. The following table describes the structure of

the request:

Path	Type	Description
`configuredLevel`	`String`	Level for the logger. May be omitted to clear the level.

16.4. Clearing a Log Level

To clear the level of a logger, make a `POST` request to `/actuator/loggers/{logger.name}` with a JSON body containing an empty object, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/loggers/com.example' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{}'
```

The preceding example clears the configured level of the `com.example` logger.

Chapter 17. Mappings (`mappings`)

The `mappings` endpoint provides information about the application's request mappings.

17.1. Retrieving the Mappings

To retrieve the mappings, make a `GET` request to `/actuator/mappings`, as shown in the following curl-based example:

```
$ curl 'http://localhost:36610/actuator/mappings' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Date: Sat, 12 Jan 2019 02:28:58 GMT
Content-Length: 5676
Transfer-Encoding: chunked

{
  "contexts" : {
    "application" : {
      "mappings" : {
        "dispatcherServlets" : {
          "dispatcherServlet" : [ {
            "handler" : "ResourceHttpRequestHandler [class path resource [META-
INF/resources/], class path resource [resources/], class path resource [static/],
class path resource [public/], ServletContext resource [/], class path resource []]",
            "predicate" : "/**/favicon.ico"
          }, {
            "handler" : "Actuator web endpoint 'mappings'",
            "predicate" : "{GET /actuator/mappings, produces [application/vnd.spring-
boot.actuator.v2+json || application/json]}",
            "details" : {
              "handlerMethod" : {
                "className" :
"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMa
pping.OperationHandler",
                "name" : "handle",
                "descriptor" :
"(Ljavax/servlet/http/HttpServletRequest;Ljava/util/Map;)Ljava/lang/Object;"
              },
              "requestMappingConditions" : {
                "consumes" : [ ],
                "headers" : [ ],
                "methods" : [ "GET" ],
                "params" : [ ],
                "patterns" : [ "/actuator/mappings" ],
```

```
                "produces" : [ {
                  "mediaType" : "application/vnd.spring-boot.actuator.v2+json",
                  "negated" : false
                }, {
                  "mediaType" : "application/json",
                  "negated" : false
                } ]
              }
            }
          }, {
            "handler" : "Actuator root web endpoint",
            "predicate" : "{GET /actuator, produces [application/vnd.spring-
boot.actuator.v2+json || application/json]}",
            "details" : {
              "handlerMethod" : {
                "className" :
"org.springframework.boot.actuate.endpoint.web.servlet.WebMvcEndpointHandlerMapping.We
bMvcLinksHandler",
                "name" : "links",
                "descriptor" :
"(Ljavax/servlet/http/HttpServletRequest;Ljavax/servlet/http/HttpServletResponse;)Ljav
a/lang/Object;"
              },
              "requestMappingConditions" : {
                "consumes" : [ ],
                "headers" : [ ],
                "methods" : [ "GET" ],
                "params" : [ ],
                "patterns" : [ "/actuator" ],
                "produces" : [ {
                  "mediaType" : "application/vnd.spring-boot.actuator.v2+json",
                  "negated" : false
                }, {
                  "mediaType" : "application/json",
                  "negated" : false
                } ]
              }
            }
          }, {
            "handler" : "public java.lang.String
org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.MappingsEndp
ointServletDocumentationTests$ExampleController.example()",
            "predicate" : "{POST /, params [a!=alpha], headers [X-Custom=Foo],
consumes [application/json || !application/xml], produces [text/plain]}",
            "details" : {
              "handlerMethod" : {
                "className" :
"org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.MappingsEnd
pointServletDocumentationTests.ExampleController",
                "name" : "example",
                "descriptor" : "()Ljava/lang/String;"
```

```
              },
              "requestMappingConditions" : {
                "consumes" : [ {
                  "mediaType" : "application/json",
                  "negated" : false
                }, {
                  "mediaType" : "application/xml",
                  "negated" : true
                } ],
                "headers" : [ {
                  "name" : "X-Custom",
                  "value" : "Foo",
                  "negated" : false
                } ],
                "methods" : [ "POST" ],
                "params" : [ {
                  "name" : "a",
                  "value" : "alpha",
                  "negated" : true
                } ],
                "patterns" : [ "/" ],
                "produces" : [ {
                  "mediaType" : "text/plain",
                  "negated" : false
                } ]
              }
            }
          }, {
            "handler" : "ResourceHttpRequestHandler [\"classpath:/META-
INF/resources/webjars/\"]",
            "predicate" : "/webjars/**"
          }, {
            "handler" : "ResourceHttpRequestHandler [\"classpath:/META-
INF/resources/\", \"classpath:/resources/\", \"classpath:/static/\",
\"classpath:/public/\", \"/\"]",
            "predicate" : "/**"
          } ]
        },
        "servletFilters" : [ {
          "servletNameMappings" : [ ],
          "urlPatternMappings" : [ "/*" ],
          "name" : "requestContextFilter",
          "className" :
"org.springframework.boot.web.servlet.filter.OrderedRequestContextFilter"
        }, {
          "servletNameMappings" : [ ],
          "urlPatternMappings" : [ "/*" ],
          "name" : "hiddenHttpMethodFilter",
          "className" :
"org.springframework.boot.web.servlet.filter.OrderedHiddenHttpMethodFilter"
        }, {
```

```
          "servletNameMappings" : [ ],
          "urlPatternMappings" : [ "/*" ],
          "name" : "formContentFilter",
          "className" :
 "org.springframework.boot.web.servlet.filter.OrderedFormContentFilter"
        } ],
        "servlets" : [ {
          "mappings" : [ ],
          "name" : "default",
          "className" : "org.apache.catalina.servlets.DefaultServlet"
        }, {
          "mappings" : [ "/" ],
          "name" : "dispatcherServlet",
          "className" : "org.springframework.web.servlet.DispatcherServlet"
        } ]
      }
    }
  }
}
```

17.1.1. Response Structure

The response contains details of the application's mappings. The items found in the response depend on the type of web application (reactive or Servlet-based). The following table describes the structure of the common elements of the response:

Path	Type	Description
`contexts`	`Object`	Application contexts keyed by id.
`contexts.*.mappings`	`Object`	Mappings in the context, keyed by mapping type.
`contexts.*.mappings.dispatcherServlets`	`Object`	Dispatcher servlet mappings, if any.
`contexts.*.mappings.servletFilters`	`Array`	Servlet filter mappings, if any.
`contexts.*.mappings.servlets`	`Array`	Servlet mappings, if any.
`contexts.*.mappings.dispatcherHandlers`	`Object`	Dispatcher handler mappings, if any.
`contexts.*.parentId`	`String`	Id of the parent application context, if any.

The entries that may be found in `contexts.*.mappings` are described in the following sections.

17.1.2. Dispatcher Servlets Response Structure

When using Spring MVC, the response contains details of any `DispatcherServlet` request mappings beneath `contexts.*.mappings.dispatcherServlets`. The following table describes the structure of this section of the response:

Path	Type	Description
`*`	`Array`	Dispatcher servlet mappings, if any, keyed by dispatcher servlet bean name.
`*.[].details`	`Object`	Additional implementation-specific details about the mapping. Optional.
`*.[].handler`	`String`	Handler for the mapping.
`*.[].predicate`	`String`	Predicate for the mapping.
`*.[].details.handlerMethod`	`Object`	Details of the method, if any, that will handle requests to this mapping.
`*.[].details.handlerMethod.className`	`Varies`	Fully qualified name of the class of the method.
`*.[].details.handlerMethod.name`	`Varies`	Name of the method.
`*.[].details.handlerMethod.descriptor`	`Varies`	Descriptor of the method as specified in the Java Language Specification.
`*.[].details.requestMappingConditions`	`Object`	Details of the request mapping conditions.
`*.[].details.requestMappingConditions.consumes`	`Varies`	Details of the consumes condition
`*.[].details.requestMappingConditions.consumes.[].mediaType`	`Varies`	Consumed media type.
`*.[].details.requestMappingConditions.consumes.[].negated`	`Varies`	Whether the media type is negated.
`*.[].details.requestMappingConditions.headers`	`Varies`	Details of the headers condition.
`*.[].details.requestMappingConditions.headers.[].name`	`Varies`	Name of the header.
`*.[].details.requestMappingConditions.headers.[].value`	`Varies`	Required value of the header, if any.
`*.[].details.requestMappingConditions.headers.[].negated`	`Varies`	Whether the value is negated.
`*.[].details.requestMappingConditions.methods`	`Varies`	HTTP methods that are handled.
`*.[].details.requestMappingConditions.params`	`Varies`	Details of the params condition.
`*.[].details.requestMappingConditions.params.[].name`	`Varies`	Name of the parameter.

Path	Type	Description
`*.[].details.requestMappingConditions.params.[].value`	`Varies`	Required value of the parameter, if any.
`*.[].details.requestMappingConditions.params.[].negated`	`Varies`	Whether the value is negated.
`*.[].details.requestMappingConditions.patterns`	`Varies`	Patterns identifying the paths handled by the mapping.
`*.[].details.requestMappingConditions.produces`	`Varies`	Details of the produces condition.
`*.[].details.requestMappingConditions.produces.[].mediaType`	`Varies`	Produced media type.
`*.[].details.requestMappingConditions.produces.[].negated`	`Varies`	Whether the media type is negated.

17.1.3. Servlets Response Structure

When using the Servlet stack, the response contains details of any `Servlet` mappings beneath `contexts.*.mappings.servlets`. The following table describes the structure of this section of the response:

Path	Type	Description
`[].mappings`	`Array`	Mappings of the servlet.
`[].name`	`String`	Name of the servlet.
`[].className`	`String`	Class name of the servlet

17.1.4. Servlet Filters Response Structure

When using the Servlet stack, the response contains details of any `Filter` mappings beneath `contexts.*.mappings.servletFilters`. The following table describes the structure of this section of the response:

Path	Type	Description
`[].servletNameMappings`	`Array`	Names of the servlets to which the filter is mapped.
`[].urlPatternMappings`	`Array`	URL pattern to which the filter is mapped.
`[].name`	`String`	Name of the filter.
`[].className`	`String`	Class name of the filter

17.1.5. Dispatcher Handlers Response Structure

When using Spring WebFlux, the response contains details of any `DispatcherHandler` request

mappings beneath `contexts.*.mappings.dispatcherHandlers`. The following table describes the structure of this section of the response:

Path	Type	Description
`*`	`Array`	Dispatcher handler mappings, if any, keyed by dispatcher handler bean name.
`*.[].details`	`Object`	Additional implementation-specific details about the mapping. Optional.
`*.[].handler`	`String`	Handler for the mapping.
`*.[].predicate`	`String`	Predicate for the mapping.
`*.[].details.requestMappingConditions`	`Object`	Details of the request mapping conditions.
`*.[].details.requestMappingConditions.consumes`	`Array`	Details of the consumes condition
`*.[].details.requestMappingConditions.consumes.[].mediaType`	`String`	Consumed media type.
`*.[].details.requestMappingConditions.consumes.[].negated`	`Boolean`	Whether the media type is negated.
`*.[].details.requestMappingConditions.headers`	`Array`	Details of the headers condition.
`*.[].details.requestMappingConditions.headers.[].name`	`String`	Name of the header.
`*.[].details.requestMappingConditions.headers.[].value`	`String`	Required value of the header, if any.
`*.[].details.requestMappingConditions.headers.[].negated`	`Boolean`	Whether the value is negated.
`*.[].details.requestMappingConditions.methods`	`Array`	HTTP methods that are handled.
`*.[].details.requestMappingConditions.params`	`Array`	Details of the params condition.
`*.[].details.requestMappingConditions.params.[].name`	`String`	Name of the parameter.
`*.[].details.requestMappingConditions.params.[].value`	`String`	Required value of the parameter, if any.
`*.[].details.requestMappingConditions.params.[].negated`	`Boolean`	Whether the value is negated.
`*.[].details.requestMappingConditions.patterns`	`Array`	Patterns identifying the paths handled by the mapping.

Path	Type	Description
`*.[].details.requestMappingConditions.produces`	`Array`	Details of the produces condition.
`*.[].details.requestMappingConditions.produces.[].mediaType`	`String`	Produced media type.
`*.[].details.requestMappingConditions.produces.[].negated`	`Boolean`	Whether the media type is negated.
`*.[].details.handlerMethod`	`Object`	Details of the method, if any, that will handle requests to this mapping.
`*.[].details.handlerMethod.className`	`String`	Fully qualified name of the class of the method.
`*.[].details.handlerMethod.name`	`String`	Name of the method.
`*.[].details.handlerMethod.descriptor`	`String`	Descriptor of the method as specified in the Java Language Specification.
`*.[].details.handlerFunction`	`Object`	Details of the function, if any, that will handle requests to this mapping.
`*.[].details.handlerFunction.className`	`String`	Fully qualified name of the class of the function.

Chapter 18. Metrics (`metrics`)

The `metrics` endpoint provides access to application metrics.

18.1. Retrieving Metric Names

To retrieve the names of the available metrics, make a `GET` request to `/actuator/metrics`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/metrics' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 154

{
  "names" : [ "jvm.memory.max", "jvm.memory.used", "jvm.memory.committed",
"jvm.buffer.memory.used", "jvm.buffer.count", "jvm.buffer.total.capacity" ]
}
```

18.1.1. Response Structure

The response contains details of the metric names. The following table describes the structure of the response:

Path	Type	Description
`names`	`Array`	Names of the known metrics.

18.2. Retrieving a Metric

To retrieve a metric, make a `GET` request to `/actuator/metrics/{metric.name}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/metrics/jvm.memory.max' -i -X GET
```

The preceding example retrieves information about the metric named `jvm.memory.max`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Disposition: inline;filename=f.txt
Content-Length: 474

{
  "name" : "jvm.memory.max",
  "description" : "The maximum amount of memory in bytes that can be used for memory
management",
  "baseUnit" : "bytes",
  "measurements" : [ {
    "statistic" : "VALUE",
    "value" : 2.381316095E9
  } ],
  "availableTags" : [ {
    "tag" : "area",
    "values" : [ "heap", "nonheap" ]
  }, {
    "tag" : "id",
    "values" : [ "Compressed Class Space", "PS Survivor Space", "PS Old Gen",
"Metaspace", "PS Eden Space", "Code Cache" ]
  } ]
}
```

18.2.1. Query Parameters

The endpoint uses query parameters to drill down into a metric by using its tags. The following table shows the single supported query parameter:

Parameter	Description
`tag`	A tag to use for drill-down in the form `name:value`.

18.2.2. Response structure

The response contains details of the metric. The following table describes the structure of the response:

Path	Type	Description
`name`	`String`	Name of the metric
`description`	`String`	Description of the metric
`baseUnit`	`String`	Base unit of the metric
`measurements`	`Array`	Measurements of the metric

Path	Type	Description
measurements[].statistic	String	Statistic of the measurement. (TOTAL, TOTAL_TIME, COUNT, MAX, VALUE, UNKNOWN, ACTIVE_TASKS, DURATION).
measurements[].value	Number	Value of the measurement.
availableTags	Array	Tags that are available for drill-down.
availableTags[].tag	String	Name of the tag.
availableTags[].values	Array	Possible values of the tag.

18.3. Drilling Down

To drill down into a metric, make a GET request to /actuator/metrics/{metric.name} using the tag query parameter, as shown in the following curl-based example:

```
$ curl
'http://localhost:8080/actuator/metrics/jvm.memory.max?tag=area%3Anonheap&tag=id%3ACom
pressed+Class+Space' -i -X GET
```

The preceding example retrieves the jvm.memory.max metric, where the area tag has a value of nonheap and the id attribute has a value of Compressed Class Space. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 263
Content-Disposition: inline;filename=f.txt

{
  "name" : "jvm.memory.max",
  "description" : "The maximum amount of memory in bytes that can be used for memory
management",
  "baseUnit" : "bytes",
  "measurements" : [ {
    "statistic" : "VALUE",
    "value" : 1.073741824E9
  } ],
  "availableTags" : [ ]
}
```

Chapter 19. Prometheus (`prometheus`)

The `prometheus` endpoint provides Spring Boot application's metrics in the format required for scraping by a Prometheus server.

19.1. Retrieving the Metrics

To retrieve the metrics, make a `GET` request to `/actuator/prometheus`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/prometheus' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Length: 2373
Content-Type: text/plain;version=0.0.4;charset=utf-8

# HELP jvm_memory_used_bytes The amount of used memory
# TYPE jvm_memory_used_bytes gauge
jvm_memory_used_bytes{area="heap",id="PS Survivor Space",} 3588760.0
jvm_memory_used_bytes{area="heap",id="PS Old Gen",} 1.73485568E8
jvm_memory_used_bytes{area="heap",id="PS Eden Space",} 5.0492304E7
jvm_memory_used_bytes{area="nonheap",id="Metaspace",} 1.50964352E8
jvm_memory_used_bytes{area="nonheap",id="Code Cache",} 5.2737216E7
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",} 2.0639328E7
# HELP jvm_memory_max_bytes The maximum amount of memory in bytes that can be used for
memory management
# TYPE jvm_memory_max_bytes gauge
jvm_memory_max_bytes{area="heap",id="PS Survivor Space",} 1.7301504E7
jvm_memory_max_bytes{area="heap",id="PS Old Gen",} 7.16177408E8
jvm_memory_max_bytes{area="heap",id="PS Eden Space",} 3.24009984E8
jvm_memory_max_bytes{area="nonheap",id="Metaspace",} -1.0
jvm_memory_max_bytes{area="nonheap",id="Code Cache",} 2.5165824E8
jvm_memory_max_bytes{area="nonheap",id="Compressed Class Space",} 1.073741824E9
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for
the Java virtual machine to use
# TYPE jvm_memory_committed_bytes gauge
jvm_memory_committed_bytes{area="heap",id="PS Survivor Space",} 1.7301504E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen",} 3.87448832E8
jvm_memory_committed_bytes{area="heap",id="PS Eden Space",} 3.20339968E8
jvm_memory_committed_bytes{area="nonheap",id="Metaspace",} 1.60428032E8
jvm_memory_committed_bytes{area="nonheap",id="Code Cache",} 5.3149696E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space",} 2.2540288E7
# HELP jvm_buffer_memory_used_bytes An estimate of the memory that the Java virtual
machine is using for this buffer pool
# TYPE jvm_buffer_memory_used_bytes gauge
jvm_buffer_memory_used_bytes{id="direct",} 802949.0
jvm_buffer_memory_used_bytes{id="mapped",} 0.0
# HELP jvm_buffer_count_buffers An estimate of the number of buffers in the pool
# TYPE jvm_buffer_count_buffers gauge
jvm_buffer_count_buffers{id="direct",} 20.0
jvm_buffer_count_buffers{id="mapped",} 0.0
# HELP jvm_buffer_total_capacity_bytes An estimate of the total capacity of the
buffers in this pool
# TYPE jvm_buffer_total_capacity_bytes gauge
jvm_buffer_total_capacity_bytes{id="direct",} 802948.0
jvm_buffer_total_capacity_bytes{id="mapped",} 0.0
```

Chapter 20. Scheduled Tasks (scheduledtasks)

The scheduledtasks endpoint provides information about the application's scheduled tasks.

20.1. Retrieving the Scheduled Tasks

To retrieve the scheduled tasks, make a GET request to /actuator/scheduledtasks, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/scheduledtasks' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 451

{
  "cron" : [ {
    "runnable" : {
      "target" : "com.example.Processor.processOrders"
    },
    "expression" : "0 0 0/3 1/1 * ?"
  } ],
  "fixedDelay" : [ {
    "runnable" : {
      "target" : "com.example.Processor.purge"
    },
    "initialDelay" : 5000,
    "interval" : 5000
  } ],
  "fixedRate" : [ {
    "runnable" : {
      "target" : "com.example.Processor.retrieveIssues"
    },
    "initialDelay" : 10000,
    "interval" : 3000
  } ]
}
```

20.1.1. Response Structure

The response contains details of the application's scheduled tasks. The following table describes the structure of the response:

Path	Type	Description
`cron`	`Array`	Cron tasks, if any.
`cron.[]runnable.target`	`String`	Target that will be executed.
`cron.[].expression`	`String`	Cron expression.
`fixedDelay`	`Array`	Fixed delay tasks, if any.
`fixedDelay.[]runnable.target`	`String`	Target that will be executed.
`fixedDelay.[].initialDelay`	`Number`	Delay, in milliseconds, before first execution.
`fixedDelay.[].interval`	`Number`	Interval, in milliseconds, between the end of the last execution and the start of the next.
`fixedRate`	`Array`	Fixed rate tasks, if any.
`fixedRate.[].runnable.target`	`String`	Target that will be executed.
`fixedRate.[].interval`	`Number`	Interval, in milliseconds, between the start of each execution.
`fixedRate.[].initialDelay`	`Number`	Delay, in milliseconds, before first execution.

Chapter 21. Sessions (`sessions`)

The `sessions` endpoint provides information about the application's HTTP sessions that are managed by Spring Session.

21.1. Retrieving Sessions

To retrieve the sessions, make a `GET` request to `/actuator/sessions`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/sessions?username=alice' -i -X GET
```

The preceding examples retrieves all of the sessions for the user whose username is `alice`.

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 753

{
  "sessions" : [ {
    "id" : "4db5efcc-99cb-4d05-a52c-b49acfbb7ea9",
    "attributeNames" : [ ],
    "creationTime" : "2019-01-11T21:29:00.858Z",
    "lastAccessedTime" : "2019-01-12T02:28:23.859Z",
    "maxInactiveInterval" : 1800,
    "expired" : false
  }, {
    "id" : "4ea24f3c-682c-4868-abaf-cc4bc165c42c",
    "attributeNames" : [ ],
    "creationTime" : "2019-01-11T14:29:00.858Z",
    "lastAccessedTime" : "2019-01-12T02:28:15.858Z",
    "maxInactiveInterval" : 1800,
    "expired" : false
  }, {
    "id" : "cb55ec30-a2ca-4d2f-9abb-0a8ed1448ad8",
    "attributeNames" : [ ],
    "creationTime" : "2019-01-12T00:29:00.859Z",
    "lastAccessedTime" : "2019-01-12T02:28:48.859Z",
    "maxInactiveInterval" : 1800,
    "expired" : false
  } ]
}
```

21.1.1. Query Parameters

The endpoint uses query parameters to limit the sessions that it returns. The following table shows the single required query parameter:

Parameter	Description
`username`	Name of the user.

21.1.2. Response Structure

The response contains details of the matching sessions. The following table describes the structure of the response:

Path	Type	Description
`sessions`	`Array`	Sessions for the given username.
`sessions.[].id`	`String`	ID of the session.
`sessions.[].attributeNames`	`Array`	Names of the attributes stored in the session.
`sessions.[].creationTime`	`String`	Timestamp of when the session was created.
`sessions.[].lastAccessedTime`	`String`	Timestamp of when the session was last accessed.
`sessions.[].maxInactiveInterval`	`Number`	Maximum permitted period of inactivity, in seconds, before the session will expire.
`sessions.[].expired`	`Boolean`	Whether the session has expired.

21.2. Retrieving a Single Session

To retrieve a single session, make a `GET` request to `/actuator/sessions/{id}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/sessions/4db5efcc-99cb-4d05-a52c-b49acfbb7ea9'
-i -X GET
```

The preceding example retrieves the session with the `id` of `4db5efcc-99cb-4d05-a52c-b49acfbb7ea9`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8
Content-Length: 228

{
  "id" : "4db5efcc-99cb-4d05-a52c-b49acfbb7ea9",
  "attributeNames" : [ ],
  "creationTime" : "2019-01-11T21:29:00.858Z",
  "lastAccessedTime" : "2019-01-12T02:28:23.859Z",
  "maxInactiveInterval" : 1800,
  "expired" : false
}
```

21.2.1. Response Structure

The response contains details of the requested session. The following table describes the structure of the response:

Path	Type	Description
`id`	`String`	ID of the session.
`attributeNames`	`Array`	Names of the attributes stored in the session.
`creationTime`	`String`	Timestamp of when the session was created.
`lastAccessedTime`	`String`	Timestamp of when the session was last accessed.
`maxInactiveInterval`	`Number`	Maximum permitted period of inactivity, in seconds, before the session will expire.
`expired`	`Boolean`	Whether the session has expired.

21.3. Deleting a Session

To delete a session, make a `DELETE` request to `/actuator/sessions/{id}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/sessions/4db5efcc-99cb-4d05-a52c-b49acfbb7ea9'
-i -X DELETE
```

The preceding example deletes the session with the `id` of `4db5efcc-99cb-4d05-a52c-b49acfbb7ea9`.

Chapter 22. Shutdown (shutdown)

The shutdown endpoint is used to shut down the application.

22.1. Shutting Down the Application

To shut down the application, make a POST request to /actuator/shutdown, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/shutdown' -i -X POST
```

A response similar to the following is produced:

```
HTTP/1.1 200 OK
Content-Length: 41
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8

{
  "message" : "Shutting down, bye..."
}
```

22.1.1. Response Structure

The response contains details of the result of the shutdown request. The following table describes the structure of the response:

Path	Type	Description
message	String	Message describing the result of the request.

Chapter 23. Thread Dump (threaddump)

The threaddump endpoint provides a thread dump from the application's JVM.

23.1. Retrieving the Thread Dump

To retrieve the thread dump, make a GET request to /actuator/threaddump, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/threaddump' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Length: 5704
Content-Type: application/vnd.spring-boot.actuator.v2+json;charset=UTF-8

{
  "threads" : [ {
    "threadName" : "Thread-134",
    "threadId" : 612,
    "blockedTime" : -1,
    "blockedCount" : 0,
    "waitedTime" : -1,
    "waitedCount" : 1,
    "lockOwnerId" : -1,
    "inNative" : false,
    "suspended" : false,
    "threadState" : "TIMED_WAITING",
    "stackTrace" : [ {
      "methodName" : "sleep",
      "fileName" : "Thread.java",
      "lineNumber" : -2,
      "className" : "java.lang.Thread",
      "nativeMethod" : true
    }, {
      "methodName" : "performShutdown",
      "fileName" : "ShutdownEndpoint.java",
      "lineNumber" : 67,
      "className" : "org.springframework.boot.actuate.context.ShutdownEndpoint",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "lineNumber" : -1,
      "className" :
"org.springframework.boot.actuate.context.ShutdownEndpoint$$Lambda$1428/2105769670",
      "nativeMethod" : false
    }, {
```

```
      "methodName" : "run",
      "fileName" : "Thread.java",
      "lineNumber" : 748,
      "className" : "java.lang.Thread",
      "nativeMethod" : false
    } ],
    "lockedMonitors" : [ ],
    "lockedSynchronizers" : [ ]
  }, {
    "threadName" : "pool-10-thread-1",
    "threadId" : 604,
    "blockedTime" : -1,
    "blockedCount" : 0,
    "waitedTime" : -1,
    "waitedCount" : 2,
    "lockName" :
"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject@91f805a",
    "lockOwnerId" : -1,
    "inNative" : false,
    "suspended" : false,
    "threadState" : "TIMED_WAITING",
    "stackTrace" : [ {
      "methodName" : "park",
      "fileName" : "Unsafe.java",
      "lineNumber" : -2,
      "className" : "sun.misc.Unsafe",
      "nativeMethod" : true
    }, {
      "methodName" : "parkNanos",
      "fileName" : "LockSupport.java",
      "lineNumber" : 215,
      "className" : "java.util.concurrent.locks.LockSupport",
      "nativeMethod" : false
    }, {
      "methodName" : "awaitNanos",
      "fileName" : "AbstractQueuedSynchronizer.java",
      "lineNumber" : 2078,
      "className" :
"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject",
      "nativeMethod" : false
    }, {
      "methodName" : "take",
      "fileName" : "ScheduledThreadPoolExecutor.java",
      "lineNumber" : 1093,
      "className" :
"java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue",
      "nativeMethod" : false
    }, {
      "methodName" : "take",
      "fileName" : "ScheduledThreadPoolExecutor.java",
      "lineNumber" : 809,
```

```
      "className" :
"java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue",
      "nativeMethod" : false
    }, {
      "methodName" : "getTask",
      "fileName" : "ThreadPoolExecutor.java",
      "lineNumber" : 1074,
      "className" : "java.util.concurrent.ThreadPoolExecutor",
      "nativeMethod" : false
    }, {
      "methodName" : "runWorker",
      "fileName" : "ThreadPoolExecutor.java",
      "lineNumber" : 1134,
      "className" : "java.util.concurrent.ThreadPoolExecutor",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : "ThreadPoolExecutor.java",
      "lineNumber" : 624,
      "className" : "java.util.concurrent.ThreadPoolExecutor$Worker",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : "Thread.java",
      "lineNumber" : 748,
      "className" : "java.lang.Thread",
      "nativeMethod" : false
    } ],
    "lockedMonitors" : [ ],
    "lockedSynchronizers" : [ ],
    "lockInfo" : {
      "className" :
"java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject",
      "identityHashCode" : 153059418
    }
  }, {
    "threadName" : "http-nio-auto-15-Acceptor-0",
    "threadId" : 597,
    "blockedTime" : -1,
    "blockedCount" : 0,
    "waitedTime" : -1,
    "waitedCount" : 0,
    "lockOwnerId" : -1,
    "inNative" : true,
    "suspended" : false,
    "threadState" : "RUNNABLE",
    "stackTrace" : [ {
      "methodName" : "accept0",
      "fileName" : "ServerSocketChannelImpl.java",
      "lineNumber" : -2,
      "className" : "sun.nio.ch.ServerSocketChannelImpl",
```

```
      "nativeMethod" : true
    }, {
      "methodName" : "accept",
      "fileName" : "ServerSocketChannelImpl.java",
      "lineNumber" : 422,
      "className" : "sun.nio.ch.ServerSocketChannelImpl",
      "nativeMethod" : false
    }, {
      "methodName" : "accept",
      "fileName" : "ServerSocketChannelImpl.java",
      "lineNumber" : 250,
      "className" : "sun.nio.ch.ServerSocketChannelImpl",
      "nativeMethod" : false
    }, {
      "methodName" : "serverSocketAccept",
      "fileName" : "NioEndpoint.java",
      "lineNumber" : 448,
      "className" : "org.apache.tomcat.util.net.NioEndpoint",
      "nativeMethod" : false
    }, {
      "methodName" : "serverSocketAccept",
      "fileName" : "NioEndpoint.java",
      "lineNumber" : 70,
      "className" : "org.apache.tomcat.util.net.NioEndpoint",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : "Acceptor.java",
      "lineNumber" : 95,
      "className" : "org.apache.tomcat.util.net.Acceptor",
      "nativeMethod" : false
    }, {
      "methodName" : "run",
      "fileName" : "Thread.java",
      "lineNumber" : 748,
      "className" : "java.lang.Thread",
      "nativeMethod" : false
    } ],
    "lockedMonitors" : [ {
      "className" : "java.lang.Object",
      "identityHashCode" : 2012491460,
      "lockedStackDepth" : 2,
      "lockedStackFrame" : {
        "methodName" : "accept",
        "fileName" : "ServerSocketChannelImpl.java",
        "lineNumber" : 250,
        "className" : "sun.nio.ch.ServerSocketChannelImpl",
        "nativeMethod" : false
      }
    } ],
    "lockedSynchronizers" : [ ]
```

```
  } ]
}
```

23.1.1. Response Structure

The response contains details of the JVM's threads. The following table describes the structure of the response:

Path	Type	Description
`threads`	`Array`	JVM's threads.
`threads.[].blockedCount`	`Number`	Total number of times that the thread has been blocked.
`threads.[].blockedTime`	`Number`	Time in milliseconds that the thread has spent blocked. -1 if thread contention monitoring is disabled.
`threads.[].daemon`	`Boolean`	Whether the thread is a daemon thread. Only available on Java 9 or later.
`threads.[].inNative`	`Boolean`	Whether the thread is executing native code.
`threads.[].lockName`	`String`	Description of the object on which the thread is blocked, if any.
`threads.[].lockInfo`	`Object`	Object for which the thread is blocked waiting.
`threads.[].lockInfo.className`	`String`	Fully qualified class name of the lock object.
`threads.[].lockInfo.identityHashCode`	`Number`	Identity hash code of the lock object.
`threads.[].lockedMonitors`	`Array`	Monitors locked by this thread, if any
`threads.[].lockedMonitors.[].className`	`String`	Class name of the lock object.
`threads.[].lockedMonitors.[].identityHashCode`	`Number`	Identity hash code of the lock object.
`threads.[].lockedMonitors.[].lockedStackDepth`	`Number`	Stack depth where the monitor was locked.
`threads.[].lockedMonitors.[].lockedStackFrame`	`Object`	Stack frame that locked the monitor.
`threads.[].lockedSynchronizers`	`Array`	Synchronizers locked by this thread.

Path	Type	Description
threads.[].lockedSynchronizers.[].className	String	Class name of the locked synchronizer.
threads.[].lockedSynchronizers.[].identifyHashCode	Number	Identity hash code of the locked synchronizer.
threads.[].lockOwnerId	Number	ID of the thread that owns the object on which the thread is blocked. -1 if the thread is not blocked.
threads.[].lockOwnerName	String	Name of the thread that owns the object on which the thread is blocked, if any.
threads.[].priority	Number	Priority of the thread. Only available on Java 9 or later.
threads.[].stackTrace	Array	Stack trace of the thread.
threads.[].stackTrace.[].classLoaderName	String	Name of the class loader of the class that contains the execution point identified by this entry, if any. Only available on Java 9 or later.
threads.[].stackTrace.[].className	String	Name of the class that contains the execution point identified by this entry.
threads.[].stackTrace.[].fileName	String	Name of the source file that contains the execution point identified by this entry, if any.
threads.[].stackTrace.[].lineNumber	Number	Line number of the execution point identified by this entry. Negative if unknown.
threads.[].stackTrace.[].methodName	String	Name of the method.
threads.[].stackTrace.[].moduleName	String	Name of the module that contains the execution point identified by this entry, if any. Only available on Java 9 or later.
threads.[].stackTrace.[].moduleVersion	String	Version of the module that contains the execution point identified by this entry, if any. Only available on Java 9 or later.
threads.[].stackTrace.[].nativeMethod	Boolean	Whether the execution point is a native method.

Path	Type	Description
`threads.[].suspended`	`Boolean`	Whether the thread is suspended.
`threads.[].threadId`	`Number`	ID of the thread.
`threads.[].threadName`	`String`	Name of the thread.
`threads.[].threadState`	`String`	State of the thread (`NEW`, `RUNNABLE`, `BLOCKED`, `WAITING`, `TIMED_WAITING`, `TERMINATED`).
`threads.[].waitedCount`	`Number`	Total number of times that the thread has waited for notification.
`threads.[].waitedTime`	`Number`	Time in milliseconds that the thread has spent waiting. -1 if thread contention monitoring is disabled