

# Spring Boot Actuator Web API Documentation

Andy Wilkinson, Stephane Nicoll

2.7.27

# Table of Contents

|                                                          |    |
|----------------------------------------------------------|----|
| 1. Overview                                              | 2  |
| 1.1. URLs                                                | 2  |
| 1.2. Timestamps                                          | 2  |
| 2. Audit Events ( <b>auditevents</b> )                   | 3  |
| 2.1. Retrieving Audit Events                             | 3  |
| 2.1.1. Query Parameters                                  | 3  |
| 2.1.2. Response Structure                                | 3  |
| 3. Beans ( <b>beans</b> )                                | 5  |
| 3.1. Retrieving the Beans                                | 5  |
| 3.1.1. Response Structure                                | 6  |
| 4. Caches ( <b>caches</b> )                              | 8  |
| 4.1. Retrieving All Caches                               | 8  |
| 4.1.1. Response Structure                                | 8  |
| 4.2. Retrieving Caches by Name                           | 9  |
| 4.2.1. Query Parameters                                  | 9  |
| 4.2.2. Response Structure                                | 9  |
| 4.3. Evict All Caches                                    | 10 |
| 4.4. Evict a Cache by Name                               | 10 |
| 4.4.1. Request Structure                                 | 10 |
| 5. Conditions Evaluation Report ( <b>conditions</b> )    | 11 |
| 5.1. Retrieving the Report                               | 11 |
| 5.1.1. Response Structure                                | 12 |
| 6. Configuration Properties ( <b>configprops</b> )       | 14 |
| 6.1. Retrieving All @ConfigurationProperties Beans       | 14 |
| 6.1.1. Response Structure                                | 16 |
| 6.2. Retrieving @ConfigurationProperties Beans By Prefix | 17 |
| 6.2.1. Response Structure                                | 19 |
| 7. Environment ( <b>env</b> )                            | 20 |
| 7.1. Retrieving the Entire Environment                   | 20 |
| 7.1.1. Response Structure                                | 21 |
| 7.2. Retrieving a Single Property                        | 22 |
| 7.2.1. Response Structure                                | 23 |
| 8. Flyway ( <b>flyway</b> )                              | 24 |
| 8.1. Retrieving the Migrations                           | 24 |
| 8.1.1. Response Structure                                | 24 |
| 9. Health ( <b>health</b> )                              | 26 |
| 9.1. Retrieving the Health of the Application            | 26 |
| 9.1.1. Response Structure                                | 27 |

|                                                                |    |
|----------------------------------------------------------------|----|
| 9.2. Retrieving the Health of a Component .....                | 28 |
| 9.2.1. Response Structure .....                                | 28 |
| 9.3. Retrieving the Health of a Nested Component .....         | 29 |
| 9.3.1. Response Structure .....                                | 29 |
| 10. Heap Dump ( <b>heapdump</b> ) .....                        | 30 |
| 10.1. Retrieving the Heap Dump .....                           | 30 |
| 11. HTTP Trace ( <b>httptrace</b> ) .....                      | 31 |
| 11.1. Retrieving the Traces .....                              | 31 |
| 11.1.1. Response Structure .....                               | 31 |
| 12. Info ( <b>info</b> ) .....                                 | 33 |
| 12.1. Retrieving the Info .....                                | 33 |
| 12.1.1. Response Structure .....                               | 33 |
| Build Response Structure .....                                 | 33 |
| Git Response Structure .....                                   | 34 |
| 13. Spring Integration graph ( <b>integrationgraph</b> ) ..... | 35 |
| 13.1. Retrieving the Spring Integration Graph .....            | 35 |
| 13.1.1. Response Structure .....                               | 36 |
| 13.2. Rebuilding the Spring Integration Graph .....            | 37 |
| 14. Liquibase ( <b>liquibase</b> ) .....                       | 38 |
| 14.1. Retrieving the Changes .....                             | 38 |
| 14.1.1. Response Structure .....                               | 38 |
| 15. Log File ( <b>logfile</b> ) .....                          | 40 |
| 15.1. Retrieving the Log File .....                            | 40 |
| 15.2. Retrieving Part of the Log File .....                    | 42 |
| 16. Loggers ( <b>loggers</b> ) .....                           | 43 |
| 16.1. Retrieving All Loggers .....                             | 43 |
| 16.1.1. Response Structure .....                               | 44 |
| 16.2. Retrieving a Single Logger .....                         | 44 |
| 16.2.1. Response Structure .....                               | 44 |
| 16.3. Retrieving a Single Group .....                          | 45 |
| 16.3.1. Response Structure .....                               | 45 |
| 16.4. Setting a Log Level .....                                | 45 |
| 16.4.1. Request Structure .....                                | 46 |
| 16.5. Setting a Log Level for a Group .....                    | 46 |
| 16.5.1. Request Structure .....                                | 46 |
| 16.6. Clearing a Log Level .....                               | 46 |
| 17. Mappings ( <b>mappings</b> ) .....                         | 47 |
| 17.1. Retrieving the Mappings .....                            | 47 |
| 17.1.1. Response Structure .....                               | 50 |
| 17.1.2. Dispatcher Servlets Response Structure .....           | 50 |
| 17.1.3. Servlets Response Structure .....                      | 52 |

|                                                              |    |
|--------------------------------------------------------------|----|
| 17.1.4. Servlet Filters Response Structure .....             | 52 |
| 17.1.5. Dispatcher Handlers Response Structure .....         | 52 |
| 18. Metrics ( <b>metrics</b> ) .....                         | 55 |
| 18.1. Retrieving Metric Names .....                          | 55 |
| 18.1.1. Response Structure .....                             | 55 |
| 18.2. Retrieving a Metric .....                              | 55 |
| 18.2.1. Query Parameters .....                               | 56 |
| 18.2.2. Response structure .....                             | 56 |
| 18.3. Drilling Down .....                                    | 57 |
| 19. Prometheus ( <b>prometheus</b> ) .....                   | 58 |
| 19.1. Retrieving All Metrics .....                           | 58 |
| 19.1.1. Query Parameters .....                               | 61 |
| 19.2. Retrieving Filtered Metrics .....                      | 62 |
| 20. Quartz ( <b>quartz</b> ) .....                           | 63 |
| 20.1. Retrieving Registered Groups .....                     | 63 |
| 20.1.1. Response Structure .....                             | 63 |
| 20.2. Retrieving Registered Job Names .....                  | 63 |
| 20.2.1. Response Structure .....                             | 64 |
| 20.3. Retrieving Registered Trigger Names .....              | 64 |
| 20.3.1. Response Structure .....                             | 65 |
| 20.4. Retrieving Overview of a Job Group .....               | 65 |
| 20.4.1. Response Structure .....                             | 66 |
| 20.5. Retrieving Overview of a Trigger Group .....           | 66 |
| 20.5.1. Response Structure .....                             | 67 |
| 20.6. Retrieving Details of a Job .....                      | 69 |
| 20.6.1. Response Structure .....                             | 70 |
| 20.7. Retrieving Details of a Trigger .....                  | 71 |
| 20.7.1. Common Response Structure .....                      | 71 |
| 20.7.2. Cron Trigger Response Structure .....                | 72 |
| 20.7.3. Simple Trigger Response Structure .....              | 73 |
| 20.7.4. Daily Time Interval Trigger Response Structure ..... | 74 |
| 20.7.5. Calendar Interval Trigger Response Structure .....   | 76 |
| 20.7.6. Custom Trigger Response Structure .....              | 77 |
| 21. Scheduled Tasks ( <b>scheduledtasks</b> ) .....          | 78 |
| 21.1. Retrieving the Scheduled Tasks .....                   | 78 |
| 21.1.1. Response Structure .....                             | 79 |
| 22. Sessions ( <b>sessions</b> ) .....                       | 80 |
| 22.1. Retrieving Sessions .....                              | 80 |
| 22.1.1. Query Parameters .....                               | 81 |
| 22.1.2. Response Structure .....                             | 81 |
| 22.2. Retrieving a Single Session .....                      | 81 |

|                                                                      |    |
|----------------------------------------------------------------------|----|
| 22.2.1. Response Structure .....                                     | 82 |
| 22.3. Deleting a Session .....                                       | 82 |
| 23. Shutdown ( <b>shutdown</b> ) .....                               | 83 |
| 23.1. Shutting Down the Application .....                            | 83 |
| 23.1.1. Response Structure .....                                     | 83 |
| 24. Application Startup ( <b>startup</b> ) .....                     | 84 |
| 24.1. Retrieving the Application Startup Steps .....                 | 84 |
| 24.1.1. Retrieving a snapshot of the Application Startup Steps ..... | 84 |
| 24.1.2. Draining the Application Startup Steps .....                 | 85 |
| 24.1.3. Response Structure .....                                     | 86 |
| 25. Thread Dump ( <b>threaddump</b> ) .....                          | 88 |
| 25.1. Retrieving the Thread Dump as JSON .....                       | 88 |
| 25.1.1. Response Structure .....                                     | 92 |
| 25.2. Retrieving the Thread Dump as Text .....                       | 95 |

This API documentation describes Spring Boot Actuators web endpoints.

# Chapter 1. Overview

Before you proceed, you should read the following topics:

- [URLs](#)
- [Timestamps](#)



In order to get the correct JSON responses documented below, Jackson must be available.

## 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=2025-07-03T08%3A56%3A49.663Z&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.v3+json
Content-Length: 121

{
  "events" : [ {
    "timestamp" : "2025-07-03T08:56:49.663Z",
    "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                                                                 |
|------------------------|-----------------------------------------------------------------------------|
| <code>after</code>     | Restricts the events to those that occurred after the given time. Optional. |
| <code>principal</code> | Restricts the events to those with the given principal. Optional.           |
| <code>type</code>      | 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.v3+json
Content-Length: 1089

{
  "contexts" : {
    "application" : {
      "beans" : {

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

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

"org.springframework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration
" : {
      "aliases" : [ ],
      "scope" : "singleton",
      "type" :
"org.springframework.boot.autoconfigure.web.servlet.DispatcherServletAutoConfiguration
",
      "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.v3+json
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                 |
|----------------------|---------------|-----------------------------|
| <b>cacheManagers</b> | <b>Object</b> | Cache managers keyed by id. |

| Path                                         | Type                | Description                                      |
|----------------------------------------------|---------------------|--------------------------------------------------|
| <code>cacheManagers.*.caches</code>          | <code>Object</code> | Caches in the application context keyed by name. |
| <code>cacheManagers.*.caches.*.target</code> | <code>String</code> | 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.v3+json
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                                                                                |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>cacheManager</code> | 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         |
|---------------------------|---------------------|---------------------|
| <code>name</code>         | <code>String</code> | Cache name.         |
| <code>cacheManager</code> | <code>String</code> | Cache manager name. |

| Path                | Type                | Description                               |
|---------------------|---------------------|-------------------------------------------|
| <code>target</code> | <code>String</code> | 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                                                                                |
|---------------------------|--------------------------------------------------------------------------------------------|
| <code>cacheManager</code> | 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.v3+json
Content-Length: 3322

{
  "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.invoke.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.PreferGsonOrJacksonAndJsonbUnavailableConditio
n.JacksonJsonbUnavailable NoneNestedConditions 1 matched 1 did not; NestedCondition on
GsonHttpMessageConvertersConfiguration.JacksonAndJsonbUnavailableCondition.JsonbPrefer
red @ConditionalOnProperty (spring.mvc.converters.preferred-json-mapper=jsonb) did not
find property 'spring.mvc.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.PreferGsonOrJacksonAndJsonbUnavailableConditio
n.GsonPreferred @ConditionalOnProperty (spring.mvc.converters.preferred-json-
mapper=gson) did not find property 'spring.mvc.converters.preferred-json-mapper'"
    } ],
    "matched" : [ ]
},

"WebMvcEndpointManagementContextConfiguration#managementHealthEndpointWebMvcHandlerMap
ping" : {
    "notMatched" : [ {
        "condition" : "OnManagementPortCondition",
        "message" : "Management Port actual port type (SAME) did not match
required type (DIFFERENT)"
    } ],
    "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 All `@ConfigurationProperties` Beans

To retrieve all of 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.v3+json
Content-Length: 3411

{
  "contexts" : {
    "application" : {
      "beans" : {
        "management.endpoints.web.cors-
org.springframework.boot.actuate.autoconfigure.endpoint.web.CorsEndpointProperties" :
{
      "prefix" : "management.endpoints.web.cors",
      "properties" : {
        "allowedOrigins" : [ ],
        "maxAge" : "PT30M",
        "exposedHeaders" : [ ],
        "allowedOriginPatterns" : [ ],
        "allowedHeaders" : [ ],
        "allowedMethods" : [ ]
      },
      "inputs" : {
        "allowedOrigins" : [ ],
        "maxAge" : { },
        "exposedHeaders" : [ ],
        "allowedOriginPatterns" : [ ],
        "allowedHeaders" : [ ],
        "allowedMethods" : [ ]
      }
    },
    "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointProperties" : {
      "prefix" : "management.endpoints.web",
```

```

    "properties" : {
      "pathMapping" : { },
      "exposure" : {
        "include" : [ "*" ],
        "exclude" : [ ]
      },
      "basePath" : "/actuator",
      "discovery" : {
        "enabled" : true
      }
    },
    "inputs" : {
      "pathMapping" : { },
      "exposure" : {
        "include" : [ {
          "value" : "*",
          "origin" : "\\management.endpoints.web.exposure.include\" from
property source \"Inlined Test Properties\""
        } ],
        "exclude" : [ ]
      },
      "basePath" : { },
      "discovery" : {
        "enabled" : { }
      }
    }
  },
  "spring.web-org.springframework.boot.autoconfigure.web.WebProperties" : {
    "prefix" : "spring.web",
    "properties" : {
      "localeResolver" : "ACCEPT_HEADER",
      "resources" : {
        "staticLocations" : [ "classpath:/META-INF/resources/",
"classpath:/resources/", "classpath:/static/", "classpath:/public/" ],
        "addMappings" : true,
        "chain" : {
          "cache" : true,
          "compressed" : false,
          "strategy" : {
            "fixed" : {
              "enabled" : false,
              "paths" : [ "/*" ]
            },
            "content" : {
              "enabled" : false,
              "paths" : [ "/*" ]
            }
          }
        }
      },
      "cache" : {
        "cachecontrol" : { },

```

```
      "useLastModified" : true  
    }  
  },  
  "inputs" : {  
    "localeResolver" : { },  
    "resources" : {  
      "staticLocations" : [ { }, { }, { }, { } ],  
      "addMappings" : { },  
      "chain" : {  
        "cache" : { },  
        "compressed" : { },  
        "strategy" : {  
          "fixed" : {  
            "enabled" : { },  
            "paths" : [ { } ]  
          },  
          "content" : {  
            "enabled" : { },  
            "paths" : [ { } ]  
          }  
        },  
        "cache" : {  
          "cachecontrol" : { },  
          "useLastModified" : { }  
        }  
      }  
    }  
  },  
  "cache" : {  
    "cachecontrol" : { },  
    "useLastModified" : { }  
  }  
}
```

### 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.           |

| Path                                   | Type                | Description                                                                    |
|----------------------------------------|---------------------|--------------------------------------------------------------------------------|
| <code>contexts.*.beans.*.inputs</code> | <code>Object</code> | Origin and value of the configuration property used when binding to this bean. |
| <code>contexts.*.parentId</code>       | <code>String</code> | Id of the parent application context, if any.                                  |

## 6.2. Retrieving @ConfigurationProperties Beans By Prefix

To retrieve the `@ConfigurationProperties` beans mapped under a certain prefix, make a `GET` request to `/actuator/configprops/{prefix}`, as shown in the following curl-based example:

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

The resulting response is similar to the following:

HTTP/1.1 200 OK

Content-Disposition: inline;filename=f.txt

Content-Type: application/vnd.spring-boot.actuator.v3+json

Content-Length: 1215

```
{
  "contexts" : {
    "application" : {
      "beans" : {
        "spring.jackson-
org.springframework.boot.autoconfigure.jackson.JacksonProperties" : {
          "prefix" : "spring.jackson",
          "properties" : {
            "serialization" : {
              "INDENT_OUTPUT" : true
            },
            "defaultPropertyInclusion" : "NON_NULL",
            "visibility" : { },
            "parser" : { },
            "deserialization" : { },
            "generator" : { },
            "mapper" : { }
          },
          "inputs" : {
            "serialization" : {
              "INDENT_OUTPUT" : {
                "value" : "true",
                "origin" : "\"spring.jackson.serialization.indent_output\" from
property source \"Inlined Test Properties\""
              }
            },
            "defaultPropertyInclusion" : {
              "value" : "non_null",
              "origin" : "\"spring.jackson.default-property-inclusion\" from property
source \"Inlined Test Properties\""
            },
            "visibility" : { },
            "parser" : { },
            "deserialization" : { },
            "generator" : { },
            "mapper" : { }
          }
        }
      }
    }
  }
}
```



The `{prefix}` does not need to be exact, a more general prefix will return all beans mapped under that prefix stem.

### 6.2.1. Response Structure

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

| Path                                       | Type                | Description                                                                    |
|--------------------------------------------|---------------------|--------------------------------------------------------------------------------|
| <code>contexts</code>                      | <code>Object</code> | Application contexts keyed by id.                                              |
| <code>contexts.*.beans.*</code>            | <code>Object</code> | <code>@ConfigurationProperties</code> beans keyed by bean name.                |
| <code>contexts.*.beans.*.prefix</code>     | <code>String</code> | Prefix applied to the names of the bean's properties.                          |
| <code>contexts.*.beans.*.properties</code> | <code>Object</code> | Properties of the bean as name-value pairs.                                    |
| <code>contexts.*.beans.*.inputs</code>     | <code>Object</code> | Origin and value of the configuration property used when binding to this bean. |
| <code>contexts.*.parentId</code>           | <code>String</code> | 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.v3+json

Content-Length: 933

```
{
  "activeProfiles" : [ ],
  "propertySources" : [ {
    "name" : "servletContextInitParams",
    "properties" : { }
  }, {
    "name" : "systemProperties",
    "properties" : {
      "java.runtime.name" : {
        "value" : "OpenJDK Runtime Environment"
      },
      "java.vm.version" : {
        "value" : "25.452-b11"
      },
      "java.vm.vendor" : {
        "value" : "BellSoft"
      }
    }
  }, {
    "name" : "systemEnvironment",
    "properties" : {
      "JAVA_HOME" : {
        "value" : "/opt/hostedtoolcache/Java_Liberica_jdk/8.0.452-11/x64",
        "origin" : "System Environment Property \"JAVA_HOME\""
      }
    }
  }, {
    "name" : "Config resource 'class path resource [application.properties]' via
location 'classpath:/'",
    "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                                               |
|----------------------------------------------------|--------|-----------------------------------------------------------|
| <code>activeProfiles</code>                        | Array  | Names of the active profiles, if any.                     |
| <code>propertySources</code>                       | Array  | Property sources in order of precedence.                  |
| <code>propertySources[].name</code>                | String | Name of the property source.                              |
| <code>propertySources[].properties</code>          | Object | Properties in the property source keyed by property name. |
| <code>propertySources[].properties.*.value</code>  | String | Value of the property.                                    |
| <code>propertySources[].properties.*.origin</code> | 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-Disposition: inline;filename=f.txt
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 564

{
  "property" : {
    "source" : "Config resource 'class path resource [application.properties]' via
location 'classpath:/'",
    "value" : "1000"
  },
  "activeProfiles" : [ ],
  "propertySources" : [ {
    "name" : "servletContextInitParams"
  }, {
    "name" : "systemProperties"
  }, {
    "name" : "systemEnvironment"
  }, {
    "name" : "Config resource 'class path resource [application.properties]' via
location 'classpath:/'",
    "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                              |
|------------------------------------------------|---------------------|------------------------------------------|
| <code>property</code>                          | <code>Object</code> | Property from the environment, if found. |
| <code>property.source</code>                   | <code>String</code> | Name of the source of the property.      |
| <code>property.value</code>                    | <code>String</code> | Value of the property.                   |
| <code>activeProfiles</code>                    | <code>Array</code>  | Names of the active profiles, if any.    |
| <code>propertySources</code>                   | <code>Array</code>  | Property sources in order of precedence. |
| <code>propertySources[].name</code>            | <code>String</code> | Name of the property source.             |
| <code>propertySources[].property</code>        | <code>Object</code> | Property in the property source, if any. |
| <code>propertySources[].property.value</code>  | <code>Varies</code> | Value of the property.                   |
| <code>propertySources[].property.origin</code> | <code>String</code> | 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.v3+json
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" : "2025-07-03T08:56:53.426Z",
            "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                                                                                                                                                                                                                         |
|------------------------------------------------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>contexts</code>                                            | Object | Application contexts keyed by id                                                                                                                                                                                                    |
| <code>contexts.*.flywayBeans.*.migrations</code>                 | Array  | Migrations performed by the Flyway instance, keyed by Flyway bean name.                                                                                                                                                             |
| <code>contexts.*.flywayBeans.*.migrations[].checksum</code>      | Number | Checksum of the migration, if any.                                                                                                                                                                                                  |
| <code>contexts.*.flywayBeans.*.migrations[].description</code>   | String | Description of the migration, if any.                                                                                                                                                                                               |
| <code>contexts.*.flywayBeans.*.migrations[].executionTime</code> | Number | Execution time in milliseconds of an applied migration.                                                                                                                                                                             |
| <code>contexts.*.flywayBeans.*.migrations[].installedBy</code>   | String | User that installed the applied migration, if any.                                                                                                                                                                                  |
| <code>contexts.*.flywayBeans.*.migrations[].installedOn</code>   | String | Timestamp of when the applied migration was installed, if any.                                                                                                                                                                      |
| <code>contexts.*.flywayBeans.*.migrations[].installedRank</code> | Number | Rank of the applied migration, if any. Later migrations have higher ranks.                                                                                                                                                          |
| <code>contexts.*.flywayBeans.*.migrations[].script</code>        | String | Name of the script used to execute the migration, if any.                                                                                                                                                                           |
| <code>contexts.*.flywayBeans.*.migrations[].state</code>         | 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, DELETED) |
| <code>contexts.*.flywayBeans.*.migrations[].type</code>          | String | Type of the migration. (SCHEMA, BASELINE, DELETE, SQL, SQL_BASELINE, UNDO_SQL, JDBC, JDBC_BASELINE, UNDO_JDBC, SCRIPT, SCRIPT_BASELINE, UNDO_SCRIPT, CUSTOM, UNDO_CUSTOM)                                                           |
| <code>contexts.*.flywayBeans.*.migrations[].version</code>       | String | Version of the database after applying the migration, if any.                                                                                                                                                                       |
| <code>contexts.*.parentId</code>                                 | 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 \
-H 'Accept: application/json'
```

The resulting response is similar to the following:

```

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 704

{
  "status" : "UP",
  "components" : {
    "broker" : {
      "status" : "UP",
      "components" : {
        "us1" : {
          "status" : "UP",
          "details" : {
            "version" : "1.0.2"
          }
        },
        "us2" : {
          "status" : "UP",
          "details" : {
            "version" : "1.0.4"
          }
        }
      }
    },
    "db" : {
      "status" : "UP",
      "details" : {
        "database" : "H2",
        "validationQuery" : "isValid()"
      }
    },
    "diskSpace" : {
      "status" : "UP",
      "details" : {
        "total" : 154894188544,
        "free" : 120272236544,
        "threshold" : 10485760,
        "exists" : true
      }
    }
  }
}

```

### 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                                                                                                                                   |
|--------------------------------------|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>status</code>                  | <code>String</code> | Overall status of the application.                                                                                                            |
| <code>components</code>              | <code>Object</code> | The components that make up the health.                                                                                                       |
| <code>components.*.status</code>     | <code>String</code> | Status of a specific part of the application.                                                                                                 |
| <code>components.*.components</code> | <code>Object</code> | The nested components that make up the health.                                                                                                |
| <code>components.*.details</code>    | <code>Object</code> | Details of the health of a specific part of the application. Presence is controlled by <code>management.endpoint.health.show-details</code> . |



The response fields above are for the V3 API. If you need to return V2 JSON you should use an accept header or `application/vnd.spring-boot.actuator.v2+json`

## 9.2. Retrieving the Health of a Component

To retrieve the health of a particular component of the application's health, 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 \
-H 'Accept: application/json'
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 101

{
  "status" : "UP",
  "details" : {
    "database" : "H2",
    "validationQuery" : "isValid()"
  }
}
```

### 9.2.1. Response Structure

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

| Path                 | Type                | Description                                                  |
|----------------------|---------------------|--------------------------------------------------------------|
| <code>status</code>  | <code>String</code> | Status of a specific part of the application                 |
| <code>details</code> | <code>Object</code> | Details of the health of a specific part of the application. |

## 9.3. Retrieving the Health of a Nested Component

If a particular component contains other nested components (as the **broker** indicator in the example above), the health of such a nested component can be retrieved by issuing a **GET** request to `/actuator/health/{component}/{subcomponent}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/health/broker/us1' -i -X GET \
-H 'Accept: application/json'
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 66

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

Components of an application's health may be nested arbitrarily deep depending on the application's health indicators and how they have been grouped. The health endpoint supports any number of `/health/{component}` identifiers in the URL to allow the health of a component at any depth to be retrieved.

### 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                                                  |
|----------------------|---------------------|--------------------------------------------------------------|
| <code>status</code>  | <code>String</code> | Status of a specific part of the application                 |
| <code>details</code> | <code>Object</code> | 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 and can be large. Its format depends upon the JVM on which the application is running. When running on a HotSpot JVM the format is `HPROF` and on OpenJ9 it is `PHD`. 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.v3+json
Content-Length: 503

{
  "traces" : [ {
    "timestamp" : "2025-07-03T08:56:54.818Z",
    "principal" : {
      "name" : "alice"
    },
    "session" : {
      "id" : "4dcb040f-6d9e-459a-bb02-fb31d97237a5"
    },
    "request" : {
      "method" : "GET",
      "uri" : "https://api.example.com",
      "headers" : {
        "Accept" : [ "application/json" ]
      }
    },
    "response" : {
      "status" : 200,
      "headers" : {
        "Content-Type" : [ "application/json" ]
      }
    },
    "timeTaken" : 0
  } ]
}
```

### 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.v3+json
Content-Length: 231

{
  "git" : {
    "commit" : {
      "time" : "2025-07-03T08:56:54Z",
      "id" : "df027cf"
    },
    "branch" : "main"
  },
  "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 several contributors that are described below.

#### Build Response Structure

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

| Path            | Type          | Description                             |
|-----------------|---------------|-----------------------------------------|
| <b>artifact</b> | <b>String</b> | Artifact ID of the application, if any. |
| <b>group</b>    | <b>String</b> | Group ID of the application, if any.    |

| Path    | Type   | Description                                          |
|---------|--------|------------------------------------------------------|
| name    | String | Name of the application, if any.                     |
| 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.          |



This is the "simple" output. The contributor can also be configured to output all available data.

# 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.v3+json
Content-Length: 962
```

```
{
  "contentDescriptor" : {
    "providerVersion" : "5.5.20",
    "providerFormatVersion" : 1.2,
    "provider" : "spring-integration"
  },
  "nodes" : [ {
    "nodeId" : 1,
    "componentType" : "null-channel",
    "integrationPatternType" : "null_channel",
    "integrationPatternCategory" : "messaging_channel",
    "properties" : { },
    "name" : "nullChannel"
  }, {
    "nodeId" : 2,
    "componentType" : "publish-subscribe-channel",
    "integrationPatternType" : "publish_subscribe_channel",
    "integrationPatternCategory" : "messaging_channel",
    "properties" : { },
    "name" : "errorChannel"
  }, {
    "nodeId" : 3,
    "componentType" : "logging-channel-adapter",
    "integrationPatternType" : "outbound_channel_adapter",
    "integrationPatternCategory" : "messaging_endpoint",
    "properties" : { },
    "input" : "errorChannel",
    "name" : "errorLogger"
  } ],
  "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.v3+json
Content-Length: 677

{
  "contexts" : {
    "application" : {
      "liquibaseBeans" : {
        "liquibase" : {
          "changeSets" : [ {
            "author" : "marceloverdijk",
            "changeLog" : "db/changelog/db.changelog-master.yaml",
            "comments" : "",
            "contexts" : [ ],
            "dateExecuted" : "2025-07-03T08:56:56.697Z",
            "deploymentId" : "1533016650",
            "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 ( <b>EXECUTED</b> , <b>FAILED</b> , <b>SKIPPED</b> , <b>RERAN</b> , <b>MARK_RAN</b> ). |
| 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-Type: text/plain;charset=UTF-8
Content-Length: 4723
```

[illegible]

```
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-Type: text/plain;charset=UTF-8
Content-Range: bytes 0-1023/4723
Content-Length: 1024

.
/\ / _ _ _ _ _ ( _ ) _ _ _ _ _ \ \ \ \
( ( ) \ _ _ | ' _ | ' _ | ' _ \ _ | \ \ \ \
\ \ _ _ ) | | _ | | | | | | ( _ | ) ) ) )
' | _ _ | . _ _ | | _ | | \ _ , | / / / /
=====|_|=====|___/=/_/_/_/
:: 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-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 791

{
  "levels" : [ "OFF", "FATAL", "ERROR", "WARN", "INFO", "DEBUG", "TRACE" ],
  "loggers" : {
    "ROOT" : {
      "configuredLevel" : "INFO",
      "effectiveLevel" : "INFO"
    },
    "com.example" : {
      "configuredLevel" : "DEBUG",
      "effectiveLevel" : "DEBUG"
    }
  },
  "groups" : {
    "test" : {
      "configuredLevel" : "INFO",
      "members" : [ "test.member1", "test.member2" ]
    },
    "web" : {
      "members" : [ "org.springframework.core.codec", "org.springframework.http",
"org.springframework.web", "org.springframework.boot.actuate.endpoint.web",
"org.springframework.boot.web.servlet.ServletContextInitializerBeans" ]
    },
    "sql" : {
      "members" : [ "org.springframework.jdbc.core", "org.hibernate.SQL",
"org.jooq.tools.LoggerListener" ]
    }
  }
}
```

### 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                                   |
|----------------------------------------|--------|-----------------------------------------------|
| <code>levels</code>                    | Array  | Levels support by the logging system.         |
| <code>loggers</code>                   | Object | Loggers keyed by name.                        |
| <code>groups</code>                    | Object | Logger groups keyed by name                   |
| <code>loggers.*.configuredLevel</code> | String | Configured level of the logger, if any.       |
| <code>loggers.*.effectiveLevel</code>  | String | Effective level of the logger.                |
| <code>groups.*.configuredLevel</code>  | String | Configured level of the logger group, if any. |
| <code>groups.*.members</code>          | Array  | Loggers that are part of this group           |

## 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-Disposition: inline;filename=f.txt
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 61

{
  "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                             |
|------------------------------|--------|-----------------------------------------|
| <code>configuredLevel</code> | String | Configured level of the logger, if any. |

| Path                        | Type                | Description                    |
|-----------------------------|---------------------|--------------------------------|
| <code>effectiveLevel</code> | <code>String</code> | Effective level of the logger. |

## 16.3. Retrieving a Single Group

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

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

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

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 82

{
  "configuredLevel" : "INFO",
  "members" : [ "test.member1", "test.member2" ]
}
```

### 16.3.1. Response Structure

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

| Path                         | Type                | Description                                   |
|------------------------------|---------------------|-----------------------------------------------|
| <code>configuredLevel</code> | <code>String</code> | Configured level of the logger group, if any. |
| <code>members</code>         | <code>Array</code>  | Loggers that are part of this group           |

## 16.4. 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`.

### 16.4.1. Request Structure

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

| Path                         | Type                | Description                                              |
|------------------------------|---------------------|----------------------------------------------------------|
| <code>configuredLevel</code> | <code>String</code> | Level for the logger. May be omitted to clear the level. |

## 16.5. Setting a Log Level for a Group

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

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

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

### 16.5.1. Request Structure

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

| Path                         | Type                | Description                                              |
|------------------------------|---------------------|----------------------------------------------------------|
| <code>configuredLevel</code> | <code>String</code> | Level for the logger. May be omitted to clear the level. |

## 16.6. 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:37127/actuator/mappings' -i -X GET \
-H 'accept-encoding: gzip' \
-H 'user-agent: ReactorNetty/1.0.48' \
-H 'accept: */*' \
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Transfer-Encoding: chunked
Date: Thu, 03 Jul 2025 08:56:59 GMT
Content-Length: 5339

{
  "contexts" : {
    "application" : {
      "mappings" : {
        "dispatcherServlets" : {
          "dispatcherServlet" : [ {
            "handler" : "Actuator web endpoint 'mappings'",
            "predicate" : "{GET [/actuator/mappings], produces
[application/vnd.spring-boot.actuator.v3+json || 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.v3+json",
        "negated" : false
    }, {
        "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.v3+json || application/vnd.spring-boot.actuator.v2+json ||
application/json]}",
    "details" : {
        "handlerMethod" : {
            "className" :
"org.springframework.boot.actuate.endpoint.web.servlet.WebMvcEndpointHandlerMapping.Web
MvcLinksHandler",
            "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.v3+json",
                "negated" : false
            }, {
                "mediaType" : "application/vnd.spring-boot.actuator.v2+json",
                "negated" : false
            }, {
                "mediaType" : "application/json",
                "negated" : false
            } ]
        }
    }
}
}, {
    "handler" :
"org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.MappingsEnd
pointServletDocumentationTests$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/], ServletContext
[/]]",
        "predicate" : "/*"
    } ]
}, {
    "servletFilters" : [ {
        "servletNameMappings" : [ ],
        "urlPatternMappings" : [ "/*" ],
        "name" : "requestContextFilter",
        "className" :
"org.springframework.boot.web.servlet.filter.OrderedRequestContextFilter"
    }, {

```

```

        "servletNameMappings" : [ ],
        "urlPatternMappings" : [ "/"* ],
        "name" : "formContentFilter",
        "className" :
"org.springframework.boot.web.servlet.filter.OrderedFormContentFilter"
    } ],
    "servlets" : [ {
        "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                                     |
|-----------------------------------------------------|--------|-------------------------------------------------|
| <code>contexts</code>                               | Object | Application contexts keyed by id.               |
| <code>contexts.*.mappings</code>                    | Object | Mappings in the context, keyed by mapping type. |
| <code>contexts.*.mappings.dispatcherServlets</code> | Object | Dispatcher servlet mappings, if any.            |
| <code>contexts.*.mappings.servletFilters</code>     | Array  | Servlet filter mappings, if any.                |
| <code>contexts.*.mappings.servlets</code>           | Array  | Servlet mappings, if any.                       |
| <code>contexts.*.mappings.dispatcherHandlers</code> | Object | Dispatcher handler mappings, if any.            |
| <code>contexts.*.parentId</code>                    | 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.v3+json
Content-Length: 154

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

### 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                 |
|--------------|--------------|-----------------------------|
| <b>names</b> | <b>Array</b> | 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-Disposition: inline;filename=f.txt
Content-Type: application/vnd.spring-boot.actuator.v3+json
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" : 2.371878911E9
  } ],
  "availableTags" : [ {
    "tag" : "area",
    "values" : [ "heap", "nonheap" ]
  }, {
    "tag" : "id",
    "values" : [ "Compressed Class Space", "PS Old Gen", "PS Survivor Space",
"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                                                       |
|------------------|-------------------------------------------------------------------|
| <code>tag</code> | A tag to use for drill-down in the form <code>name:value</code> . |

### 18.2.2. Response structure

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

| Path                      | Type                | Description                |
|---------------------------|---------------------|----------------------------|
| <code>name</code>         | <code>String</code> | Name of the metric         |
| <code>description</code>  | <code>String</code> | Description of the metric  |
| <code>baseUnit</code>     | <code>String</code> | Base unit of the metric    |
| <code>measurements</code> | <code>Array</code>  | Measurements of the metric |

| Path                                  | Type   | Description                                                                                            |
|---------------------------------------|--------|--------------------------------------------------------------------------------------------------------|
| <code>measurements[].statistic</code> | String | Statistic of the measurement. (TOTAL, TOTAL_TIME, COUNT, MAX, VALUE, UNKNOWN, ACTIVE_TASKS, DURATION). |
| <code>measurements[].value</code>     | Number | Value of the measurement.                                                                              |
| <code>availableTags</code>            | Array  | Tags that are available for drill-down.                                                                |
| <code>availableTags[].tag</code>      | String | Name of the tag.                                                                                       |
| <code>availableTags[].values</code>   | 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%3Aanonheap&tag=id%3ACompressed+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-Disposition: inline;filename=f.txt
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 263

{
  "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 All Metrics

To retrieve all 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-Type: text/plain;version=0.0.4;charset=utf-8

Content-Length: 2372

```
# 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",} 262144.0
jvm_buffer_total_capacity_bytes{id="mapped",} 0.0
# 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",} 2.5690112E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen",} 2.05520896E8
jvm_memory_committed_bytes{area="heap",id="PS Eden Space",} 3.02514176E8
jvm_memory_committed_bytes{area="nonheap",id="Metaspace",} 9.1488256E7
jvm_memory_committed_bytes{area="nonheap",id="Code Cache",} 2.4313856E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space",} 1.2845056E7
# 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",} 262145.0
jvm_buffer_memory_used_bytes{id="mapped",} 0.0
# 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",} 1.5753352E7
jvm_memory_used_bytes{area="heap",id="PS Old Gen",} 5.752468E7
jvm_memory_used_bytes{area="heap",id="PS Eden Space",} 2.50296304E8
jvm_memory_used_bytes{area="nonheap",id="Metaspace",} 8.5232832E7
jvm_memory_used_bytes{area="nonheap",id="Code Cache",} 2.4007744E7
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",} 1.1653392E7
# 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",} 2.5690112E7
jvm_memory_max_bytes{area="heap",id="PS Old Gen",} 7.16177408E8
jvm_memory_max_bytes{area="heap",id="PS Eden Space",} 3.05135616E8
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_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",} 10.0
jvm_buffer_count_buffers{id="mapped",} 0.0
```

The default response content type is `text/plain;version=0.0.4`. The endpoint can also produce `application/openmetrics-text;version=1.0.0` when called with an appropriate `Accept` header, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/prometheus' -i -X GET \  
-H 'Accept: application/openmetrics-text; version=1.0.0; charset=utf-8'
```

The resulting response is similar to the following:

HTTP/1.1 200 OK

Content-Type: application/openmetrics-text;version=1.0.0;charset=utf-8

Content-Length: 2354

```
# TYPE jvm_buffer_total_capacity_bytes gauge
# HELP jvm_buffer_total_capacity_bytes An estimate of the total capacity of the
buffers in this pool
jvm_buffer_total_capacity_bytes{id="direct"} 262144.0
jvm_buffer_total_capacity_bytes{id="mapped"} 0.0
# TYPE jvm_memory_committed_bytes gauge
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for
the Java virtual machine to use
jvm_memory_committed_bytes{area="heap",id="PS Survivor Space"} 2.5690112E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen"} 2.05520896E8
jvm_memory_committed_bytes{area="heap",id="PS Eden Space"} 3.02514176E8
jvm_memory_committed_bytes{area="nonheap",id="Metaspace"} 9.1488256E7
jvm_memory_committed_bytes{area="nonheap",id="Code Cache"} 2.4313856E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space"} 1.2845056E7
# TYPE jvm_buffer_memory_used_bytes gauge
# HELP jvm_buffer_memory_used_bytes An estimate of the memory that the Java virtual
machine is using for this buffer pool
jvm_buffer_memory_used_bytes{id="direct"} 262145.0
jvm_buffer_memory_used_bytes{id="mapped"} 0.0
# TYPE jvm_memory_used_bytes gauge
# HELP jvm_memory_used_bytes The amount of used memory
jvm_memory_used_bytes{area="heap",id="PS Survivor Space"} 1.5753352E7
jvm_memory_used_bytes{area="heap",id="PS Old Gen"} 5.752468E7
jvm_memory_used_bytes{area="heap",id="PS Eden Space"} 2.44607816E8
jvm_memory_used_bytes{area="nonheap",id="Metaspace"} 8.5209944E7
jvm_memory_used_bytes{area="nonheap",id="Code Cache"} 2.3987648E7
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space"} 1.1650432E7
# TYPE jvm_memory_max_bytes gauge
# HELP jvm_memory_max_bytes The maximum amount of memory in bytes that can be used for
memory management
jvm_memory_max_bytes{area="heap",id="PS Survivor Space"} 2.5690112E7
jvm_memory_max_bytes{area="heap",id="PS Old Gen"} 7.16177408E8
jvm_memory_max_bytes{area="heap",id="PS Eden Space"} 3.05135616E8
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
# TYPE jvm_buffer_count_buffers gauge
# HELP jvm_buffer_count_buffers An estimate of the number of buffers in the pool
jvm_buffer_count_buffers{id="direct"} 10.0
jvm_buffer_count_buffers{id="mapped"} 0.0
# EOF
```

### 19.1.1. Query Parameters

The endpoint uses query parameters to limit the samples that it returns. The following table shows

the supported query parameters:

| Parameter                  | Description                                                    |
|----------------------------|----------------------------------------------------------------|
| <code>includedNames</code> | Restricts the samples to those that match the names. Optional. |

## 19.2. Retrieving Filtered Metrics

To retrieve metrics matching specific names, make a `GET` request to `/actuator/prometheus` with the `includedNames` query parameter, as shown in the following curl-based example:

```
$ curl
'http://localhost:8080/actuator/prometheus?includedNames=jvm_memory_used_bytes%2Cjvm_m
emory_committed_bytes' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: text/plain;version=0.0.4;charset=utf-8
Content-Length: 1108

# 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",} 2.5690112E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen",} 2.05520896E8
jvm_memory_committed_bytes{area="heap",id="PS Eden Space",} 3.02514176E8
jvm_memory_committed_bytes{area="nonheap",id="Metaspace",} 9.1488256E7
jvm_memory_committed_bytes{area="nonheap",id="Code Cache",} 2.4313856E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space",} 1.2845056E7
# 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",} 1.5753352E7
jvm_memory_used_bytes{area="heap",id="PS Old Gen",} 5.752468E7
jvm_memory_used_bytes{area="heap",id="PS Eden Space",} 2.50296304E8
jvm_memory_used_bytes{area="nonheap",id="Metaspace",} 8.5247504E7
jvm_memory_used_bytes{area="nonheap",id="Code Cache",} 2.4013696E7
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",} 1.1655088E7
```

# Chapter 20. Quartz (quartz)

The **quartz** endpoint provides information about jobs and triggers that are managed by the Quartz Scheduler.

## 20.1. Retrieving Registered Groups

Jobs and triggers are managed in groups. To retrieve the list of registered job and trigger groups, make a **GET** request to **/actuator/quartz**, as shown in the following curl-based example:

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

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 120

{
  "jobs" : {
    "groups" : [ "samples", "tests" ]
  },
  "triggers" : {
    "groups" : [ "samples", "DEFAULT" ]
  }
}
```

### 20.1.1. Response Structure

The response contains the groups names for registered jobs and triggers. The following table describes the structure of the response:

| Path                   | Type         | Description                      |
|------------------------|--------------|----------------------------------|
| <b>jobs.groups</b>     | <b>Array</b> | An array of job group names.     |
| <b>triggers.groups</b> | <b>Array</b> | An array of trigger group names. |

## 20.2. Retrieving Registered Job Names

To retrieve the list of registered job names, make a **GET** request to **/actuator/quartz/jobs**, as shown in the following curl-based example:

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

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 137
```

```
{
  "groups" : {
    "samples" : {
      "jobs" : [ "jobOne", "jobTwo" ]
    },
    "tests" : {
      "jobs" : [ "jobThree" ]
    }
  }
}
```

### 20.2.1. Response Structure

The response contains the registered job names for each group. The following table describes the structure of the response:

| Path                       | Type                | Description               |
|----------------------------|---------------------|---------------------------|
| <code>groups</code>        | <code>Object</code> | Job groups keyed by name. |
| <code>groups.*.jobs</code> | <code>Array</code>  | An array of job names.    |

## 20.3. Retrieving Registered Trigger Names

To retrieve the list of registered trigger names, make a `GET` request to `/actuator/quartz/triggers`, as shown in the following curl-based example:

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

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 229
```

```
{
  "groups" : {
    "samples" : {
      "paused" : false,
      "triggers" : [ "3am-weekdays", "every-day", "once-a-week" ]
    },
    "DEFAULT" : {
      "paused" : false,
      "triggers" : [ "every-hour-tue-thu" ]
    }
  }
}
```

### 20.3.1. Response Structure

The response contains the registered trigger names for each group. The following table describes the structure of the response:

| Path                           | Type                 | Description                           |
|--------------------------------|----------------------|---------------------------------------|
| <code>groups</code>            | <code>Object</code>  | Trigger groups keyed by name.         |
| <code>groups.*.paused</code>   | <code>Boolean</code> | Whether this trigger group is paused. |
| <code>groups.*.triggers</code> | <code>Array</code>   | An array of trigger names.            |

## 20.4. Retrieving Overview of a Job Group

To retrieve an overview of the jobs in a particular group, make a `GET` request to `/actuator/quartz/jobs/{groupName}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/quartz/jobs/samples' -i -X GET
```

The preceding example retrieves the summary for jobs in the `samples` group. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 201

{
  "group" : "samples",
  "jobs" : {
    "jobOne" : {
      "className" : "org.springframework.scheduling.quartz.DelegatingJob"
    },
    "jobTwo" : {
      "className" : "org.quartz.Job"
    }
  }
}
```

### 20.4.1. Response Structure

The response contains an overview of jobs in a particular group. The following table describes the structure of the response:

| Path                          | Type                | Description                                     |
|-------------------------------|---------------------|-------------------------------------------------|
| <code>group</code>            | <code>String</code> | Name of the group.                              |
| <code>jobs</code>             | <code>Object</code> | Job details keyed by name.                      |
| <code>jobs.*.className</code> | <code>String</code> | Fully qualified name of the job implementation. |

## 20.5. Retrieving Overview of a Trigger Group

To retrieve an overview of the triggers in a particular group, make a `GET` request to `/actuator/quartz/triggers/{groupName}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/quartz/triggers/tests' -i -X GET
```

The preceding example retrieves the summary for triggers in the `tests` group. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 1268

{
  "group" : "tests",
  "paused" : false,
  "triggers" : {
```

```

    "cron" : {
      "3am-week" : {
        "previousFireTime" : "2020-12-04T03:00:00.000+00:00",
        "nextFireTime" : "2020-12-07T03:00:00.000+00:00",
        "priority" : 3,
        "expression" : "0 0 3 ? * 1,2,3,4,5",
        "timeZone" : "Europe/Paris"
      }
    },
    "simple" : {
      "every-day" : {
        "nextFireTime" : "2020-12-04T12:00:00.000+00:00",
        "priority" : 7,
        "interval" : 86400000
      }
    },
    "dailyTimeInterval" : {
      "tue-thu" : {
        "priority" : 5,
        "interval" : 3600000,
        "daysOfWeek" : [ 3, 5 ],
        "startTimeOfDay" : "09:00:00",
        "endTimeOfDay" : "18:00:00"
      }
    },
    "calendarInterval" : {
      "once-a-week" : {
        "previousFireTime" : "2020-12-02T14:00:00.000+00:00",
        "nextFireTime" : "2020-12-08T14:00:00.000+00:00",
        "priority" : 5,
        "interval" : 604800000,
        "timeZone" : "Etc/UTC"
      }
    },
    "custom" : {
      "once-a-year-custom" : {
        "previousFireTime" : "2020-07-14T16:00:00.000+00:00",
        "nextFireTime" : "2021-07-14T16:00:00.000+00:00",
        "priority" : 10,
        "trigger" : "com.example.CustomTrigger@fdsfsd"
      }
    }
  }
}

```

### 20.5.1. Response Structure

The response contains an overview of triggers in a particular group. Trigger implementation specific details are available. The following table describes the structure of the response:

| Path                                          | Type    | Description                                                                                                  |
|-----------------------------------------------|---------|--------------------------------------------------------------------------------------------------------------|
| group                                         | String  | Name of the group.                                                                                           |
| paused                                        | Boolean | Whether the group is paused.                                                                                 |
| triggers.cron                                 | Object  | Cron triggers keyed by name, if any.                                                                         |
| triggers.simple                               | Object  | Simple triggers keyed by name, if any.                                                                       |
| triggers.dailyTimeInterval                    | Object  | Daily time interval triggers keyed by name, if any.                                                          |
| triggers.calendarInterval                     | Object  | Calendar interval triggers keyed by name, if any.                                                            |
| triggers.custom                               | Object  | Any other triggers keyed by name, if any.                                                                    |
| triggers.cron.*.previousFireTime              | String  | Last time the trigger fired, if any.                                                                         |
| triggers.cron.*.nextFireTime                  | String  | Next time at which the Trigger is scheduled to fire, if any.                                                 |
| triggers.cron.*.priority                      | Number  | Priority to use if two triggers have the same scheduled fire time.                                           |
| triggers.cron.*.expression                    | String  | Cron expression to use.                                                                                      |
| triggers.cron.*.timeZone                      | String  | Time zone for which the expression will be resolved, if any.                                                 |
| triggers.simple.*.previousFireTime            | String  | Last time the trigger fired, if any.                                                                         |
| triggers.simple.*.nextFireTime                | String  | Next time at which the Trigger is scheduled to fire, if any.                                                 |
| triggers.simple.*.priority                    | Number  | Priority to use if two triggers have the same scheduled fire time.                                           |
| triggers.simple.*.interval                    | Number  | Interval, in milliseconds, between two executions.                                                           |
| triggers.dailyTimeInterval.*.previousFireTime | String  | Last time the trigger fired, if any.                                                                         |
| triggers.dailyTimeInterval.*.nextFireTime     | String  | Next time at which the Trigger is scheduled to fire, if any.                                                 |
| triggers.dailyTimeInterval.*.priority         | Number  | Priority to use if two triggers have the same scheduled fire time.                                           |
| triggers.dailyTimeInterval.*.interval         | Number  | Interval, in milliseconds, added to the fire time in order to calculate the time of the next trigger repeat. |
| triggers.dailyTimeInterval.*.daysOfWeek       | Array   | An array of days of the week upon which to fire.                                                             |
| triggers.dailyTimeInterval.*.startTimeOfDay   | String  | Time of day to start firing at the given interval, if any.                                                   |

| Path                                                      | Type   | Description                                                                                                  |
|-----------------------------------------------------------|--------|--------------------------------------------------------------------------------------------------------------|
| <code>triggers.dailyTimeInterval.*.endTimeOfDay</code>    | String | Time of day to complete firing at the given interval, if any.                                                |
| <code>triggers.calendarInterval.*.previousFireTime</code> | String | Last time the trigger fired, if any.                                                                         |
| <code>triggers.calendarInterval.*.nextFireTime</code>     | String | Next time at which the Trigger is scheduled to fire, if any.                                                 |
| <code>triggers.calendarInterval.*.priority</code>         | Number | Priority to use if two triggers have the same scheduled fire time.                                           |
| <code>triggers.calendarInterval.*.interval</code>         | Number | Interval, in milliseconds, added to the fire time in order to calculate the time of the next trigger repeat. |
| <code>triggers.calendarInterval.*.timeZone</code>         | String | Time zone within which time calculations will be performed, if any.                                          |
| <code>triggers.custom.*.previousFireTime</code>           | String | Last time the trigger fired, if any.                                                                         |
| <code>triggers.custom.*.nextFireTime</code>               | String | Next time at which the Trigger is scheduled to fire, if any.                                                 |
| <code>triggers.custom.*.priority</code>                   | Number | Priority to use if two triggers have the same scheduled fire time.                                           |
| <code>triggers.custom.*.trigger</code>                    | String | A toString representation of the custom trigger instance.                                                    |

## 20.6. Retrieving Details of a Job

To retrieve the details about a particular job, make a **GET** request to `/actuator/quartz/jobs/{groupName}/{jobName}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/quartz/jobs/samples/jobOne' -i -X GET
```

The preceding example retrieves the details of the job identified by the `samples` group and `jobOne` name. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 609

{
  "group" : "samples",
  "name" : "jobOne",
  "description" : "A sample job",
  "className" : "org.springframework.scheduling.quartz.DelegatingJob",
  "durable" : false,
  "requestRecovery" : false,
  "data" : {
    "password" : "*****",
    "user" : "admin"
  },
  "triggers" : [ {
    "group" : "samples",
    "name" : "every-day",
    "previousFireTime" : "2020-12-04T03:00:00.000+00:00",
    "nextFireTime" : "2020-12-04T12:00:00.000+00:00",
    "priority" : 7
  }, {
    "group" : "samples",
    "name" : "3am-weekdays",
    "nextFireTime" : "2020-12-07T03:00:00.000+00:00",
    "priority" : 3
  } ]
}
```

If a key in the data map is identified as sensitive, its value is sanitized.

### 20.6.1. Response Structure

The response contains the full details of a job including a summary of the triggers associated with it, if any. The triggers are sorted by next fire time and priority. The following table describes the structure of the response:

| Path            | Type    | Description                                                                                    |
|-----------------|---------|------------------------------------------------------------------------------------------------|
| group           | String  | Name of the group.                                                                             |
| name            | String  | Name of the job.                                                                               |
| description     | String  | Description of the job, if any.                                                                |
| className       | String  | Fully qualified name of the job implementation.                                                |
| durable         | Boolean | Whether the job should remain stored after it is orphaned.                                     |
| requestRecovery | Boolean | Whether the job should be re-executed if a 'recovery' or 'fail-over' situation is encountered. |

| Path                                     | Type   | Description                                                        |
|------------------------------------------|--------|--------------------------------------------------------------------|
| <code>data.*</code>                      | String | Job data map as key/value pairs, if any.                           |
| <code>triggers</code>                    | Array  | An array of triggers associated to the job, if any.                |
| <code>triggers[].group</code>            | String | Name of the trigger group.                                         |
| <code>triggers[].name</code>             | String | Name of the trigger.                                               |
| <code>triggers[].previousFireTime</code> | String | Last time the trigger fired, if any.                               |
| <code>triggers[].nextFireTime</code>     | String | Next time at which the Trigger is scheduled to fire, if any.       |
| <code>triggers[].priority</code>         | Number | Priority to use if two triggers have the same scheduled fire time. |

## 20.7. Retrieving Details of a Trigger

To retrieve the details about a particular trigger, make a `GET` request to `/actuator/quartz/triggers/{groupName}/{triggerName}`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/quartz/triggers/samples/example' -i -X GET
```

The preceding example retrieves the details of trigger identified by the `samples` group and `example` name.

### 20.7.1. Common Response Structure

The response has a common structure and an additional object that is specific to the trigger's type. There are five supported types:

- `cron` for `CronTrigger`
- `simple` for `SimpleTrigger`
- `dailyTimeInterval` for `DailyTimeIntervalTrigger`
- `calendarInterval` for `CalendarIntervalTrigger`
- `custom` for any other trigger implementations

The following table describes the structure of the common elements of the response:

| Path                     | Type   | Description                                                                                                                                                 |
|--------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>group</code>       | String | Name of the group.                                                                                                                                          |
| <code>name</code>        | String | Name of the trigger.                                                                                                                                        |
| <code>description</code> | String | Description of the trigger, if any.                                                                                                                         |
| <code>state</code>       | String | State of the trigger ( <code>NONE</code> , <code>NORMAL</code> , <code>PAUSED</code> , <code>COMPLETE</code> , <code>ERROR</code> , <code>BLOCKED</code> ). |

| Path                           | Type   | Description                                                                                                                                                                                                                |
|--------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>type</code>              | String | Type of the trigger ( <code>calendarInterval</code> , <code>cron</code> , <code>custom</code> , <code>dailyTimeInterval</code> , <code>simple</code> ). Determines the key of the object containing type-specific details. |
| <code>calendarName</code>      | String | Name of the Calendar associated with this Trigger, if any.                                                                                                                                                                 |
| <code>startTime</code>         | String | Time at which the Trigger should take effect, if any.                                                                                                                                                                      |
| <code>endTime</code>           | String | Time at which the Trigger should quit repeating, regardless of any remaining repeats, if any.                                                                                                                              |
| <code>previousFireTime</code>  | String | Last time the trigger fired, if any.                                                                                                                                                                                       |
| <code>nextFireTime</code>      | String | Next time at which the Trigger is scheduled to fire, if any.                                                                                                                                                               |
| <code>priority</code>          | Number | Priority to use if two triggers have the same scheduled fire time.                                                                                                                                                         |
| <code>finalFireTime</code>     | String | Last time at which the Trigger will fire, if any.                                                                                                                                                                          |
| <code>data</code>              | Object | Job data map keyed by name, if any.                                                                                                                                                                                        |
| <code>calendarInterval</code>  | Object | Calendar time interval trigger details, if any. Present when <code>type</code> is <code>calendarInterval</code> .                                                                                                          |
| <code>custom</code>            | Object | Custom trigger details, if any. Present when <code>type</code> is <code>custom</code> .                                                                                                                                    |
| <code>cron</code>              | Object | Cron trigger details, if any. Present when <code>type</code> is <code>cron</code> .                                                                                                                                        |
| <code>dailyTimeInterval</code> | Object | Daily time interval trigger details, if any. Present when <code>type</code> is <code>dailyTimeInterval</code> .                                                                                                            |
| <code>simple</code>            | Object | Simple trigger details, if any. Present when <code>type</code> is <code>simple</code> .                                                                                                                                    |

### 20.7.2. Cron Trigger Response Structure

A cron trigger defines the cron expression that is used to determine when it has to fire. The resulting response for such a trigger implementation is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 490
```

```
{
  "group" : "samples",
  "name" : "example",
  "description" : "Example trigger",
  "state" : "NORMAL",
  "type" : "cron",
  "calendarName" : "bankHolidays",
  "startTime" : "2020-11-30T17:00:00.000+00:00",
  "endTime" : "2020-12-30T03:00:00.000+00:00",
  "previousFireTime" : "2020-12-04T03:00:00.000+00:00",
  "nextFireTime" : "2020-12-07T03:00:00.000+00:00",
  "priority" : 3,
  "data" : { },
  "cron" : {
    "expression" : "0 0 3 ? * 1,2,3,4,5",
    "timeZone" : "Europe/Paris"
  }
}
```

Much of the response is common to all trigger types. The structure of the common elements of the response was [described previously](#). The following table describes the structure of the parts of the response that are specific to cron triggers:

| Path                         | Type                | Description                                                  |
|------------------------------|---------------------|--------------------------------------------------------------|
| <code>cron</code>            | <code>Object</code> | Cron trigger specific details.                               |
| <code>cron.expression</code> | <code>String</code> | Cron expression to use.                                      |
| <code>cron.timeZone</code>   | <code>String</code> | Time zone for which the expression will be resolved, if any. |

### 20.7.3. Simple Trigger Response Structure

A simple trigger is used to fire a Job at a given moment in time, and optionally repeated at a specified interval. The resulting response for such a trigger implementation is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 549
```

```
{
  "group" : "samples",
  "name" : "example",
  "description" : "Example trigger",
  "state" : "NORMAL",
  "type" : "simple",
  "calendarName" : "bankHolidays",
  "startTime" : "2020-11-30T17:00:00.000+00:00",
  "endTime" : "2020-12-30T03:00:00.000+00:00",
  "previousFireTime" : "2020-12-04T03:00:00.000+00:00",
  "nextFireTime" : "2020-12-07T03:00:00.000+00:00",
  "priority" : 7,
  "finalFireTime" : "2020-12-29T17:00:00.000+00:00",
  "data" : { },
  "simple" : {
    "interval" : 86400000,
    "repeatCount" : -1,
    "timesTriggered" : 0
  }
}
```

Much of the response is common to all trigger types. The structure of the common elements of the response was [described previously](#). The following table describes the structure of the parts of the response that are specific to simple triggers:

| Path                               | Type   | Description                                                              |
|------------------------------------|--------|--------------------------------------------------------------------------|
| <code>simple</code>                | Object | Simple trigger specific details.                                         |
| <code>simple.interval</code>       | Number | Interval, in milliseconds, between two executions.                       |
| <code>simple.repeatCount</code>    | Number | Number of times the trigger should repeat, or -1 to repeat indefinitely. |
| <code>simple.timesTriggered</code> | Number | Number of times the trigger has already fired.                           |

#### 20.7.4. Daily Time Interval Trigger Response Structure

A daily time interval trigger is used to fire a Job based upon daily repeating time intervals. The resulting response for such a trigger implementation is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 667
```

```
{
  "group" : "samples",
  "name" : "example",
  "description" : "Example trigger",
  "state" : "PAUSED",
  "type" : "dailyTimeInterval",
  "calendarName" : "bankHolidays",
  "startTime" : "2020-11-30T17:00:00.000+00:00",
  "endTime" : "2020-12-30T03:00:00.000+00:00",
  "previousFireTime" : "2020-12-04T03:00:00.000+00:00",
  "nextFireTime" : "2020-12-07T03:00:00.000+00:00",
  "priority" : 5,
  "finalFireTime" : "2020-12-30T18:00:00.000+00:00",
  "data" : { },
  "dailyTimeInterval" : {
    "interval" : 3600000,
    "daysOfWeek" : [ 3, 5 ],
    "startTimeOfDay" : "09:00:00",
    "endTimeOfDay" : "18:00:00",
    "repeatCount" : -1,
    "timesTriggered" : 0
  }
}
```

Much of the response is common to all trigger types. The structure of the common elements of the response was [described previously](#). The following table describes the structure of the parts of the response that are specific to daily time interval triggers:

| Path                                          | Type   | Description                                                                                                  |
|-----------------------------------------------|--------|--------------------------------------------------------------------------------------------------------------|
| <code>dailyTimeInterval</code>                | Object | Daily time interval trigger specific details.                                                                |
| <code>dailyTimeInterval.interval</code>       | Number | Interval, in milliseconds, added to the fire time in order to calculate the time of the next trigger repeat. |
| <code>dailyTimeInterval.daysOfWeek</code>     | Array  | An array of days of the week upon which to fire.                                                             |
| <code>dailyTimeInterval.startTimeOfDay</code> | String | Time of day to start firing at the given interval, if any.                                                   |
| <code>dailyTimeInterval.endTimeOfDay</code>   | String | Time of day to complete firing at the given interval, if any.                                                |
| <code>dailyTimeInterval.repeatCount</code>    | Number | Number of times the trigger should repeat, or -1 to repeat indefinitely.                                     |
| <code>dailyTimeInterval.timesTriggered</code> | Number | Number of times the trigger has already fired.                                                               |

## 20.7.5. Calendar Interval Trigger Response Structure

A calendar interval trigger is used to fire a Job based upon repeating calendar time intervals. The resulting response for such a trigger implementation is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 669

{
  "group" : "samples",
  "name" : "example",
  "description" : "Example trigger",
  "state" : "NORMAL",
  "type" : "calendarInterval",
  "calendarName" : "bankHolidays",
  "startTime" : "2020-11-30T17:00:00.000+00:00",
  "endTime" : "2020-12-30T03:00:00.000+00:00",
  "previousFireTime" : "2020-12-04T03:00:00.000+00:00",
  "nextFireTime" : "2020-12-07T03:00:00.000+00:00",
  "priority" : 5,
  "finalFireTime" : "2020-12-28T17:00:00.000+00:00",
  "data" : { },
  "calendarInterval" : {
    "interval" : 604800000,
    "timeZone" : "Etc/UTC",
    "timesTriggered" : 0,
    "preserveHourOfDayAcrossDaylightSavings" : false,
    "skipDayIfHourDoesNotExist" : false
  }
}
```

Much of the response is common to all trigger types. The structure of the common elements of the response was [described previously](#). The following table describes the structure of the parts of the response that are specific to calendar interval triggers:

| Path                                         | Type   | Description                                                                                                  |
|----------------------------------------------|--------|--------------------------------------------------------------------------------------------------------------|
| <code>calendarInterval</code>                | Object | Calendar interval trigger specific details.                                                                  |
| <code>calendarInterval.interval</code>       | Number | Interval, in milliseconds, added to the fire time in order to calculate the time of the next trigger repeat. |
| <code>calendarInterval.timeZone</code>       | String | Time zone within which time calculations will be performed, if any.                                          |
| <code>calendarInterval.timesTriggered</code> | Number | Number of times the trigger has already fired.                                                               |

| Path                                                                 | Type    | Description                                                                                          |
|----------------------------------------------------------------------|---------|------------------------------------------------------------------------------------------------------|
| <code>calendarInterval.preserveHourOfDayAcrossDaylightSavings</code> | Boolean | Whether to fire the trigger at the same time of day, regardless of daylight saving time transitions. |
| <code>calendarInterval.skipDayIfHourDoesNotExist</code>              | Boolean | Whether to skip if the hour of the day does not exist on a given day.                                |

### 20.7.6. Custom Trigger Response Structure

A custom trigger is any other implementation. The resulting response for such a trigger implementation is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 457

{
  "group" : "samples",
  "name" : "example",
  "description" : "Example trigger.",
  "state" : "NORMAL",
  "type" : "custom",
  "calendarName" : "bankHolidays",
  "startTime" : "2020-11-30T17:00:00.000+00:00",
  "endTime" : "2020-12-30T03:00:00.000+00:00",
  "previousFireTime" : "2020-12-04T03:00:00.000+00:00",
  "nextFireTime" : "2020-12-07T03:00:00.000+00:00",
  "priority" : 10,
  "custom" : {
    "trigger" : "com.example.CustomTrigger@fdsfsd"
  }
}
```

Much of the response is common to all trigger types. The structure of the common elements of the response was [described previously](#). The following table describes the structure of the parts of the response that are specific to custom triggers:

| Path                        | Type   | Description                                               |
|-----------------------------|--------|-----------------------------------------------------------|
| <code>custom</code>         | Object | Custom trigger specific details.                          |
| <code>custom.trigger</code> | String | A toString representation of the custom trigger instance. |

# Chapter 21. Scheduled Tasks (scheduledtasks)

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

## 21.1. 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.v3+json
Content-Length: 629

{
  "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
  } ],
  "custom" : [ {
    "runnable" : {
      "target" : "com.example.Processor$CustomTriggeredRunnable"
    },
    "trigger" : "com.example.Processor$CustomTrigger@6b5f91a5"
  } ]
}
```

### 21.1.1. Response Structure

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

| Path                                      | Type   | Description                                                                                 |
|-------------------------------------------|--------|---------------------------------------------------------------------------------------------|
| <code>cron</code>                         | Array  | Cron tasks, if any.                                                                         |
| <code>cron[].runnable.target</code>       | String | Target that will be executed.                                                               |
| <code>cron[].expression</code>            | String | Cron expression.                                                                            |
| <code>fixedDelay</code>                   | Array  | Fixed delay tasks, if any.                                                                  |
| <code>fixedDelay[].runnable.target</code> | String | Target that will be executed.                                                               |
| <code>fixedDelay[].initialDelay</code>    | Number | Delay, in milliseconds, before first execution.                                             |
| <code>fixedDelay[].interval</code>        | Number | Interval, in milliseconds, between the end of the last execution and the start of the next. |
| <code>fixedRate</code>                    | Array  | Fixed rate tasks, if any.                                                                   |
| <code>fixedRate[].runnable.target</code>  | String | Target that will be executed.                                                               |
| <code>fixedRate[].interval</code>         | Number | Interval, in milliseconds, between the start of each execution.                             |
| <code>fixedRate[].initialDelay</code>     | Number | Delay, in milliseconds, before first execution.                                             |
| <code>custom</code>                       | Array  | Tasks with custom triggers, if any.                                                         |
| <code>custom[].runnable.target</code>     | String | Target that will be executed.                                                               |
| <code>custom[].trigger</code>             | String | Trigger for the task.                                                                       |

# Chapter 22. Sessions (sessions)

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

## 22.1. 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.v3+json
Content-Length: 753

{
  "sessions" : [ {
    "id" : "98f7cd18-64cd-4f93-9ce6-d7282ef31840",
    "attributeNames" : [ ],
    "creationTime" : "2025-07-02T20:57:02.011Z",
    "lastAccessedTime" : "2025-07-03T08:56:17.011Z",
    "maxInactiveInterval" : 1800,
    "expired" : false
  }, {
    "id" : "403e3c42-74f2-4569-a284-0118fef4b6d1",
    "attributeNames" : [ ],
    "creationTime" : "2025-07-03T06:57:02.011Z",
    "lastAccessedTime" : "2025-07-03T08:56:50.011Z",
    "maxInactiveInterval" : 1800,
    "expired" : false
  }, {
    "id" : "4db5efcc-99cb-4d05-a52c-b49acfb7ea9",
    "attributeNames" : [ ],
    "creationTime" : "2025-07-03T03:57:02.011Z",
    "lastAccessedTime" : "2025-07-03T08:56:25.011Z",
    "maxInactiveInterval" : 1800,
    "expired" : false
  } ]
}
```

### 22.1.1. 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       |
|-----------------------|-------------------|
| <code>username</code> | Name of the user. |

### 22.1.2. Response Structure

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

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

## 22.2. 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-b49acfb7ea9'
-i -X GET
```

The preceding example retrieves the session with the `id` of `4db5efcc-99cb-4d05-a52c-b49acfb7ea9`. The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 228

{
  "id" : "4db5efcc-99cb-4d05-a52c-b49acfbb7ea9",
  "attributeNames" : [ ],
  "creationTime" : "2025-07-03T03:57:02.011Z",
  "lastAccessedTime" : "2025-07-03T08:56:25.011Z",
  "maxInactiveInterval" : 1800,
  "expired" : false
}
```

### 22.2.1. Response Structure

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

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

## 22.3. 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 23. Shutdown (shutdown)

The **shutdown** endpoint is used to shut down the application.

## 23.1. 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-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 41

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

### 23.1.1. 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                                   |
|----------------|---------------|-----------------------------------------------|
| <b>message</b> | <b>String</b> | Message describing the result of the request. |

# Chapter 24. Application Startup (startup)

The `startup` endpoint provides information about the application's startup sequence.

## 24.1. Retrieving the Application Startup Steps

The application startup steps can either be retrieved as a snapshot (`GET`) or drained from the buffer (`POST`).

### 24.1.1. Retrieving a snapshot of the Application Startup Steps

To retrieve the steps recorded so far during the application startup phase, make a `GET` request to `/actuator/startup`, as shown in the following curl-based example:

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

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 840

{
  "springBootVersion" : "2.7.27",
  "timeline" : {
    "startTime" : "2025-07-03T08:57:03.020Z",
    "events" : [ {
      "endTime" : "2025-07-03T08:57:03.380Z",
      "duration" : "PT0S",
      "startTime" : "2025-07-03T08:57:03.380Z",
      "startupStep" : {
        "name" : "spring.beans.instantiate",
        "id" : 3,
        "tags" : [ {
          "key" : "beanName",
          "value" : "homeController"
        } ],
        "parentId" : 2
      }
    }, {
      "endTime" : "2025-07-03T08:57:03.380Z",
      "duration" : "PT0S",
      "startTime" : "2025-07-03T08:57:03.380Z",
      "startupStep" : {
        "name" : "spring.boot.application.starting",
        "id" : 2,
        "tags" : [ {
          "key" : "mainApplicationClass",
          "value" : "com.example.startup.StartupApplication"
        } ]
      }
    } ]
  }
}
```

### 24.1.2. Draining the Application Startup Steps

To drain and return the steps recorded so far during the application startup phase, make a **POST** request to `/actuator/startup`, as shown in the following curl-based example:

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

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 844
```

```
{
  "springBootVersion" : "2.7.27",
  "timeline" : {
    "startTime" : "2025-07-03T08:57:03.020Z",
    "events" : [ {
      "endTime" : "2025-07-03T08:57:03.366Z",
      "duration" : "PT0S",
      "startTime" : "2025-07-03T08:57:03.366Z",
      "startupStep" : {
        "name" : "spring.beans.instantiate",
        "id" : 1,
        "tags" : [ {
          "key" : "beanName",
          "value" : "homeController"
        } ],
        "parentId" : 0
      }
    }, {
      "endTime" : "2025-07-03T08:57:03.366Z",
      "duration" : "PT0.001S",
      "startTime" : "2025-07-03T08:57:03.365Z",
      "startupStep" : {
        "name" : "spring.boot.application.starting",
        "id" : 0,
        "tags" : [ {
          "key" : "mainApplicationClass",
          "value" : "com.example.startup.StartupApplication"
        } ]
      }
    }
  ]
}
```

### 24.1.3. Response Structure

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

| Path                            | Type                | Description                                                    |
|---------------------------------|---------------------|----------------------------------------------------------------|
| <code>springBootVersion</code>  | <code>String</code> | Spring Boot version for this application.                      |
| <code>timeline.startTime</code> | <code>String</code> | Start time of the application.                                 |
| <code>timeline.events</code>    | <code>Array</code>  | An array of steps collected during application startup so far. |

| Path                                                    | Type   | Description                                            |
|---------------------------------------------------------|--------|--------------------------------------------------------|
| <code>timeline.events[].startTime</code>                | String | The timestamp of the start of this event.              |
| <code>timeline.events[].endTime</code>                  | String | The timestamp of the end of this event.                |
| <code>timeline.events[].duration</code>                 | String | The precise duration of this event.                    |
| <code>timeline.events[].startupStep.name</code>         | String | The name of the StartupStep.                           |
| <code>timeline.events[].startupStep.id</code>           | Number | The id of this StartupStep.                            |
| <code>timeline.events[].startupStep.parentId</code>     | Number | The parent id for this StartupStep.                    |
| <code>timeline.events[].startupStep.tags</code>         | Array  | An array of key/value pairs with additional step info. |
| <code>timeline.events[].startupStep.tags[].key</code>   | String | The key of the StartupStep Tag.                        |
| <code>timeline.events[].startupStep.tags[].value</code> | String | The value of the StartupStep Tag.                      |

# Chapter 25. Thread Dump (threaddump)

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

## 25.1. Retrieving the Thread Dump as JSON

To retrieve the thread dump as JSON, make a `GET` request to `/actuator/threaddump` with an appropriate `Accept` header, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/threaddump' -i -X GET \
-H 'Accept: application/json'
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 6935

{
  "threads" : [ {
    "threadName" : "Thread-35",
    "threadId" : 68,
    "blockedTime" : -1,
    "blockedCount" : 0,
    "waitedTime" : -1,
    "waitedCount" : 1,
    "lockName" : "java.util.concurrent.CountDownLatch$Sync@64514009",
    "lockOwnerId" : -1,
    "inNative" : false,
    "suspended" : false,
    "threadState" : "WAITING",
    "stackTrace" : [ {
      "methodName" : "park",
      "fileName" : "Unsafe.java",
      "lineNumber" : -2,
      "className" : "sun.misc.Unsafe",
      "nativeMethod" : true
    }, {
      "methodName" : "park",
      "fileName" : "LockSupport.java",
      "lineNumber" : 175,
      "className" : "java.util.concurrent.locks.LockSupport",
      "nativeMethod" : false
    }, {
      "methodName" : "parkAndCheckInterrupt",
      "fileName" : "AbstractQueuedSynchronizer.java",
      "lineNumber" : 836,
      "className" : "java.util.concurrent.locks.AbstractQueuedSynchronizer",
```

```

    "nativeMethod" : false
  }, {
    "methodName" : "doAcquireSharedInterruptibly",
    "fileName" : "AbstractQueuedSynchronizer.java",
    "lineNumber" : 997,
    "className" : "java.util.concurrent.locks.AbstractQueuedSynchronizer",
    "nativeMethod" : false
  }, {
    "methodName" : "acquireSharedInterruptibly",
    "fileName" : "AbstractQueuedSynchronizer.java",
    "lineNumber" : 1304,
    "className" : "java.util.concurrent.locks.AbstractQueuedSynchronizer",
    "nativeMethod" : false
  }, {
    "methodName" : "await",
    "fileName" : "CountDownLatch.java",
    "lineNumber" : 231,
    "className" : "java.util.concurrent.CountDownLatch",
    "nativeMethod" : false
  }, {
    "methodName" : "lambda$jsonThreadDump$0",
    "fileName" : "ThreadDumpEndpointDocumentationTests.java",
    "lineNumber" : 56,
    "className" :
"org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.ThreadDumpE
ndpointDocumentationTests",
    "nativeMethod" : false
  }, {
    "methodName" : "run",
    "lineNumber" : -1,
    "className" :
"org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.ThreadDumpE
ndpointDocumentationTests$$Lambda$1759/571551137",
    "nativeMethod" : false
  }, {
    "methodName" : "run",
    "fileName" : "Thread.java",
    "lineNumber" : 750,
    "className" : "java.lang.Thread",
    "nativeMethod" : false
  } ],
  "lockedMonitors" : [ ],
  "lockedSynchronizers" : [ {
    "className" : "java.util.concurrent.locks.ReentrantLock$NonfairSync",
    "identityHashCode" : 1325793389
  } ],
  "lockInfo" : {
    "className" : "java.util.concurrent.CountDownLatch$Sync",
    "identityHashCode" : 1683046409
  }
}, {

```

```

"threadName" : "Thread-34",
"threadId" : 67,
"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" : 65,
  "className" : "org.springframework.boot.actuate.context.ShutdownEndpoint",
  "nativeMethod" : false
}, {
  "methodName" : "run",
  "lineNumber" : -1,
  "className" :
"org.springframework.boot.actuate.context.ShutdownEndpoint$$Lambda$1748/1596134131",
  "nativeMethod" : false
}, {
  "methodName" : "run",
  "fileName" : "Thread.java",
  "lineNumber" : 750,
  "className" : "java.lang.Thread",
  "nativeMethod" : false
} ],
"lockedMonitors" : [ ],
"lockedSynchronizers" : [ ]
}, {
  "threadName" : "pool-5-thread-1",
  "threadId" : 61,
  "blockedTime" : -1,
  "blockedCount" : 0,
  "waitedTime" : -1,
  "waitedCount" : 0,
  "lockOwnerId" : -1,
  "inNative" : false,
  "suspended" : false,
  "threadState" : "RUNNABLE",
  "stackTrace" : [ {
    "methodName" : "schedule",
    "fileName" : "ReschedulingRunnable.java",

```

```

    "lineNumber" : 76,
    "className" : "org.springframework.scheduling.concurrent.ReschedulingRunnable",
    "nativeMethod" : false
  }, {
    "methodName" : "run",
    "fileName" : "ReschedulingRunnable.java",
    "lineNumber" : 101,
    "className" : "org.springframework.scheduling.concurrent.ReschedulingRunnable",
    "nativeMethod" : false
  }, {
    "methodName" : "call",
    "fileName" : "Executors.java",
    "lineNumber" : 511,
    "className" : "java.util.concurrent.Executors$RunnableAdapter",
    "nativeMethod" : false
  }, {
    "methodName" : "run",
    "fileName" : "FutureTask.java",
    "lineNumber" : 266,
    "className" : "java.util.concurrent.FutureTask",
    "nativeMethod" : false
  }, {
    "methodName" : "access$201",
    "fileName" : "ScheduledThreadPoolExecutor.java",
    "lineNumber" : 180,
    "className" :
"java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask",
    "nativeMethod" : false
  }, {
    "methodName" : "run",
    "fileName" : "ScheduledThreadPoolExecutor.java",
    "lineNumber" : 293,
    "className" :
"java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask",
    "nativeMethod" : false
  }, {
    "methodName" : "runWorker",
    "fileName" : "ThreadPoolExecutor.java",
    "lineNumber" : 1149,
    "className" : "java.util.concurrent.ThreadPoolExecutor",
    "nativeMethod" : false
  }, {
    "methodName" : "run",
    "fileName" : "ThreadPoolExecutor.java",
    "lineNumber" : 624,
    "className" : "java.util.concurrent.ThreadPoolExecutor$Worker",
    "nativeMethod" : false
  }, {
    "methodName" : "run",
    "fileName" : "Thread.java",
    "lineNumber" : 750,

```

```

        "className" : "java.lang.Thread",
        "nativeMethod" : false
    } ],
    "lockedMonitors" : [ {
        "className" : "java.lang.Object",
        "identityHashCode" : 1642004219,
        "lockedStackFrame" : {
            "methodName" : "schedule",
            "fileName" : "ReschedulingRunnable.java",
            "lineNumber" : 76,
            "className" :
"org.springframework.scheduling.concurrent.ReschedulingRunnable",
            "nativeMethod" : false
        },
        "lockedStackDepth" : 0
    }, {
        "className" : "java.lang.Object",
        "identityHashCode" : 1642004219,
        "lockedStackFrame" : {
            "methodName" : "run",
            "fileName" : "ReschedulingRunnable.java",
            "lineNumber" : 101,
            "className" :
"org.springframework.scheduling.concurrent.ReschedulingRunnable",
            "nativeMethod" : false
        },
        "lockedStackDepth" : 1
    } ],
    "lockedSynchronizers" : [ {
        "className" : "java.util.concurrent.ThreadPoolExecutor$Worker",
        "identityHashCode" : 2104370022
    } ]
    } ]
}

```

### 25.1.1. Response Structure

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

| Path                                | Type   | Description                                                                                             |
|-------------------------------------|--------|---------------------------------------------------------------------------------------------------------|
| <code>threads</code>                | Array  | JVM's threads.                                                                                          |
| <code>threads[].blockedCount</code> | Number | Total number of times that the thread has been blocked.                                                 |
| <code>threads[].blockedTime</code>  | Number | Time in milliseconds that the thread has spent blocked. -1 if thread contention monitoring is disabled. |

| Path                                                          | Type    | Description                                                                                                         |
|---------------------------------------------------------------|---------|---------------------------------------------------------------------------------------------------------------------|
| <code>threads[].daemon</code>                                 | Boolean | Whether the thread is a daemon thread. Only available on Java 9 or later.                                           |
| <code>threads[].inNative</code>                               | Boolean | Whether the thread is executing native code.                                                                        |
| <code>threads[].lockName</code>                               | String  | Description of the object on which the thread is blocked, if any.                                                   |
| <code>threads[].lockInfo</code>                               | Object  | Object for which the thread is blocked waiting.                                                                     |
| <code>threads[].lockInfo.className</code>                     | String  | Fully qualified class name of the lock object.                                                                      |
| <code>threads[].lockInfo.identityHashCode</code>              | Number  | Identity hash code of the lock object.                                                                              |
| <code>threads[].lockedMonitors</code>                         | Array   | Monitors locked by this thread, if any                                                                              |
| <code>threads[].lockedMonitors[].className</code>             | String  | Class name of the lock object.                                                                                      |
| <code>threads[].lockedMonitors[].identityHashCode</code>      | Number  | Identity hash code of the lock object.                                                                              |
| <code>threads[].lockedMonitors[].lockedStackDepth</code>      | Number  | Stack depth where the monitor was locked.                                                                           |
| <code>threads[].lockedMonitors[].lockedStackFrame</code>      | Object  | Stack frame that locked the monitor.                                                                                |
| <code>threads[].lockedSynchronizers</code>                    | Array   | Synchronizers locked by this thread.                                                                                |
| <code>threads[].lockedSynchronizers[].className</code>        | String  | Class name of the locked synchronizer.                                                                              |
| <code>threads[].lockedSynchronizers[].identityHashCode</code> | Number  | Identity hash code of the locked synchronizer.                                                                      |
| <code>threads[].lockOwnerId</code>                            | Number  | ID of the thread that owns the object on which the thread is blocked. <code>-1</code> if the thread is not blocked. |
| <code>threads[].lockOwnerName</code>                          | String  | Name of the thread that owns the object on which the thread is blocked, if any.                                     |
| <code>threads[].priority</code>                               | Number  | Priority of the thread. Only available on Java 9 or later.                                                          |
| <code>threads[].stackTrace</code>                             | Array   | Stack trace of the thread.                                                                                          |

| Path                                                | Type    | Description                                                                                                                                  |
|-----------------------------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>threads[].stackTrace[].classLoaderName</code> | 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. |
| <code>threads[].stackTrace[].className</code>       | String  | Name of the class that contains the execution point identified by this entry.                                                                |
| <code>threads[].stackTrace[].fileName</code>        | String  | Name of the source file that contains the execution point identified by this entry, if any.                                                  |
| <code>threads[].stackTrace[].lineNumber</code>      | Number  | Line number of the execution point identified by this entry. Negative if unknown.                                                            |
| <code>threads[].stackTrace[].methodName</code>      | String  | Name of the method.                                                                                                                          |
| <code>threads[].stackTrace[].moduleName</code>      | String  | Name of the module that contains the execution point identified by this entry, if any. Only available on Java 9 or later.                    |
| <code>threads[].stackTrace[].moduleVersion</code>   | String  | Version of the module that contains the execution point identified by this entry, if any. Only available on Java 9 or later.                 |
| <code>threads[].stackTrace[].nativeMethod</code>    | Boolean | Whether the execution point is a native method.                                                                                              |
| <code>threads[].suspended</code>                    | Boolean | Whether the thread is suspended.                                                                                                             |
| <code>threads[].threadId</code>                     | Number  | ID of the thread.                                                                                                                            |
| <code>threads[].threadName</code>                   | String  | Name of the thread.                                                                                                                          |
| <code>threads[].threadState</code>                  | String  | State of the thread (NEW, RUNNABLE, BLOCKED, WAITING, TIMED_WAITING, TERMINATED).                                                            |
| <code>threads[].waitedCount</code>                  | Number  | Total number of times that the thread has waited for notification.                                                                           |
| <code>threads[].waitedTime</code>                   | Number  | Time in milliseconds that the thread has spent waiting. -1 if thread contention monitoring is disabled                                       |

## 25.2. Retrieving the Thread Dump as Text

To retrieve the thread dump as text, make a **GET** request to `/actuator/threaddump` that accepts `text/plain`, as shown in the following curl-based example:

```
$ curl 'http://localhost:8080/actuator/threaddump' -i -X GET \  
-H 'Accept: text/plain'
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK  
Content-Type: text/plain;charset=UTF-8  
Content-Length: 43463  
  
2025-07-03 08:57:03  
Full thread dump OpenJDK 64-Bit Server VM (25.452-b11 mixed mode):  
  
"Thread-34" - Thread t@67  
  java.lang.Thread.State: TIMED_WAITING  
    at java.lang.Thread.sleep(Native Method)  
    at  
org.springframework.boot.actuate.context.ShutdownEndpoint.performShutdown(ShutdownEndp  
oint.java:65)  
    at  
org.springframework.boot.actuate.context.ShutdownEndpoint$$Lambda$1748/1596134131.run(  
Unknown Source)  
    at java.lang.Thread.run(Thread.java:750)  
  
  Locked ownable synchronizers:  
    - None  
  
"pool-5-thread-1" - Thread t@61  
  java.lang.Thread.State: RUNNABLE  
    at  
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.siftUp(ScheduledThre  
adPoolExecutor.java:886)  
    at  
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.offer(ScheduledThrea  
dPoolExecutor.java:1020)  
    at  
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.add(ScheduledThreadP  
oolExecutor.java:1037)  
    at  
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.add(ScheduledThreadP  
oolExecutor.java:809)  
    at  
java.util.concurrent.ScheduledThreadPoolExecutor.delayedExecute(ScheduledThreadPoolExe  
cutor.java:328)  
    at
```

```

java.util.concurrent.ScheduledThreadPoolExecutor.schedule(ScheduledThreadPoolExecutor.
java:533)
    at
java.util.concurrent.Executors$DelegatedScheduledExecutorService.schedule(Executors.ja
va:729)
    at
org.springframework.scheduling.concurrent.ReschedulingRunnable.schedule(ReschedulingRu
nnable.java:82)
    - locked <61defefb> (a java.lang.Object)
    at
org.springframework.scheduling.concurrent.ReschedulingRunnable.run(ReschedulingRunnabl
e.java:101)
    - locked <61defefb> (a java.lang.Object)
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$201(Schedu
ledThreadPoolExecutor.java:180)
    at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThre
adPoolExecutor.java:293)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:750)

```

Locked ownable synchronizers:

- Locked <648100de> (a java.util.concurrent.locks.ReentrantLock\$NonfairSync)
- Locked <7d6e2366> (a java.util.concurrent.ThreadPoolExecutor\$Worker)

"http-nio-auto-1-Acceptor" - Thread t@56

java.lang.Thread.State: RUNNABLE

```

at sun.nio.ch.ServerSocketChannelImpl.accept0(Native Method)
at sun.nio.ch.ServerSocketChannelImpl.accept(ServerSocketChannelImpl.java:421)
at sun.nio.ch.ServerSocketChannelImpl.accept(ServerSocketChannelImpl.java:249)
    - locked <676b85f3> (a java.lang.Object)
at org.apache.tomcat.util.net.NioEndpoint.serverSocketAccept(NioEndpoint.java:568)
at org.apache.tomcat.util.net.NioEndpoint.serverSocketAccept(NioEndpoint.java:71)
at org.apache.tomcat.util.net.Acceptor.run(Acceptor.java:128)
at java.lang.Thread.run(Thread.java:750)

```

Locked ownable synchronizers:

- None

"http-nio-auto-1-Poller" - Thread t@55

java.lang.Thread.State: RUNNABLE

```

at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
at sun.nio.ch.EPollArrayWrapper.poll(EPollArrayWrapper.java:269)
at sun.nio.ch.EPollSelectorImpl.doSelect(EPollSelectorImpl.java:93)
at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
    - locked <42fa1554> (a sun.nio.ch.Util$3)
    - locked <544502d6> (a java.util.Collections$UnmodifiableSet)

```

```
- locked <382e7f70> (a sun.nio.ch.EPollSelectorImpl)
at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:97)
at org.apache.tomcat.util.net.NioEndpoint$Poller.run(NioEndpoint.java:833)
at java.lang.Thread.run(Thread.java:750)
```

Locked ownable synchronizers:

- None

"http-nio-auto-1-exec-10" - Thread t@54

java.lang.Thread.State: WAITING

at sun.misc.Unsafe.park(Native Method)

- parking to wait for <3d1319b> (a

java.util.concurrent.locks.AbstractQueuedSynchronizer\$ConditionObject)

at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)

at

java.util.concurrent.locks.AbstractQueuedSynchronizer\$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)

at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)

at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)

at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)

at

org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113)

at

org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1174)

at

org.apache.tomcat.util.threads.ThreadPoolExecutor\$Worker.run(ThreadPoolExecutor.java:658)

at

org.apache.tomcat.util.threads.TaskThread\$WrappingRunnable.run(TaskThread.java:63)

at java.lang.Thread.run(Thread.java:750)

Locked ownable synchronizers:

- None

"http-nio-auto-1-exec-9" - Thread t@53

java.lang.Thread.State: WAITING

at sun.misc.Unsafe.park(Native Method)

- parking to wait for <3d1319b> (a

java.util.concurrent.locks.AbstractQueuedSynchronizer\$ConditionObject)

at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)

at

java.util.concurrent.locks.AbstractQueuedSynchronizer\$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)

at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)

at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)

at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)

at

org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113)

```

    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:11
74)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:6
58)
    at
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
    at java.lang.Thread.run(Thread.java:750)

Locked ownable synchronizers:
- None

"http-nio-auto-1-exec-8" - Thread t@52
  java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <3d1319b> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQu
euedSynchronizer.java:2039)
    at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113
)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:11
74)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:6
58)
    at
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
    at java.lang.Thread.run(Thread.java:750)

Locked ownable synchronizers:
- None

"http-nio-auto-1-exec-7" - Thread t@51
  java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <3d1319b> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQu
euedSynchronizer.java:2039)
    at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)

```

```

    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113
)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:11
74)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:6
58)
    at
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
    at java.lang.Thread.run(Thread.java:750)

```

Locked ownable synchronizers:  
- None

```

"http-nio-auto-1-exec-6" - Thread t@50
  java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <3d1319b> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQu
euedSynchronizer.java:2039)
    at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113
)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:11
74)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:6
58)
    at
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
    at java.lang.Thread.run(Thread.java:750)

```

Locked ownable synchronizers:  
- None

```

"http-nio-auto-1-exec-5" - Thread t@49
  java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <3d1319b> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)

```

```

    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
    at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1174)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:658)
    at
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
    at java.lang.Thread.run(Thread.java:750)

```

Locked ownable synchronizers:

- None

"http-nio-auto-1-exec-4" - Thread t@48

java.lang.Thread.State: WAITING

at sun.misc.Unsafe.park(Native Method)

- parking to wait for <3d1319b> (a

java.util.concurrent.locks.AbstractQueuedSynchronizer\$ConditionObject)

at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)

at

```

java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
    at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)
    at

```

```

org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113)
    at

```

```

org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1174)
    at

```

```

org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:658)
    at

```

```

org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
    at java.lang.Thread.run(Thread.java:750)

```

Locked ownable synchronizers:

- None

```

"http-nio-auto-1-exec-3" - Thread t@47
  java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
      - parking to wait for <3d1319b> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
    at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1174)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:658)
    at
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
    at java.lang.Thread.run(Thread.java:750)

Locked ownable synchronizers:
- None

"http-nio-auto-1-exec-2" - Thread t@46
  java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
      - parking to wait for <3d1319b> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
    at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)
    at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1174)
    at
org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:658)
    at
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)

```

```
at java.lang.Thread.run(Thread.java:750)
```

Locked ownable synchronizers:

- None

"http-nio-auto-1-exec-1" - Thread t@45

java.lang.Thread.State: WAITING

at sun.misc.Unsafe.park(Native Method)

- parking to wait for <3d1319b> (a

java.util.concurrent.locks.AbstractQueuedSynchronizer\$ConditionObject)

at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)

at

java.util.concurrent.locks.AbstractQueuedSynchronizer\$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)

at java.util.concurrent.LinkedBlockingQueue.take(LinkedBlockingQueue.java:442)

at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:120)

at org.apache.tomcat.util.threads.TaskQueue.take(TaskQueue.java:33)

at

org.apache.tomcat.util.threads.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1113)

at

org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1174)

at

org.apache.tomcat.util.threads.ThreadPoolExecutor\$Worker.run(ThreadPoolExecutor.java:658)

at

org.apache.tomcat.util.threads.TaskThread\$WrappingRunnable.run(TaskThread.java:63)

at java.lang.Thread.run(Thread.java:750)

Locked ownable synchronizers:

- None

"container-0" - Thread t@44

java.lang.Thread.State: TIMED\_WAITING

at java.lang.Thread.sleep(Native Method)

at org.apache.catalina.core.StandardServer.await(StandardServer.java:524)

at

org.springframework.boot.web.embedded.tomcat.TomcatWebServer\$1.run(TomcatWebServer.java:197)

Locked ownable synchronizers:

- None

"Catalina-utility-2" - Thread t@43

java.lang.Thread.State: WAITING

at sun.misc.Unsafe.park(Native Method)

- parking to wait for <1193d400> (a

java.util.concurrent.locks.AbstractQueuedSynchronizer\$ConditionObject)

at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)

at

```
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
    at
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1088)
    at
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:809)
    at java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1074)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1134)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
    at java.lang.Thread.run(Thread.java:750)
```

Locked ownable synchronizers:  
- None

```
"Catalina-utility-1" - Thread t@42
  java.lang.Thread.State: TIMED_WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <1193d400> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2078)
    at
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1093)
    at
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:809)
    at java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1074)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1134)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at
org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:63)
    at java.lang.Thread.run(Thread.java:750)
```

Locked ownable synchronizers:  
- None

```
"boundedElastic-1" - Thread t@40
  java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <6dbc3756> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
```

```
euedSynchronizer.java:2039)
```

```
at
```

```
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThread  
PoolExecutor.java:1081)
```

```
at
```

```
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThread  
PoolExecutor.java:809)
```

```
at java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1074)
```

```
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1134)
```

```
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
```

```
at java.lang.Thread.run(Thread.java:750)
```

```
Locked ownable synchronizers:
```

```
- None
```

```
"reactor-http-epoll-4" - Thread t@39
```

```
java.lang.Thread.State: RUNNABLE
```

```
at io.netty.channel.epoll.Native.epollWait(Native Method)
```

```
at io.netty.channel.epoll.Native.epollWait(Native.java:220)
```

```
at io.netty.channel.epoll.Native.epollWait(Native.java:213)
```

```
at
```

```
io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:308)
```

```
at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:365)
```

```
at
```

```
io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.jav  
a:998)
```

```
at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)
```

```
at
```

```
io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
```

```
at java.lang.Thread.run(Thread.java:750)
```

```
Locked ownable synchronizers:
```

```
- None
```

```
"reactor-http-epoll-3" - Thread t@38
```

```
java.lang.Thread.State: RUNNABLE
```

```
at io.netty.channel.epoll.Native.epollWait(Native Method)
```

```
at io.netty.channel.epoll.Native.epollWait(Native.java:220)
```

```
at io.netty.channel.epoll.Native.epollWait(Native.java:213)
```

```
at
```

```
io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:308)
```

```
at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:365)
```

```
at
```

```
io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.jav  
a:998)
```

```
at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)
```

```
at
```

```
io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
```

```
at java.lang.Thread.run(Thread.java:750)
```

```
Locked ownable synchronizers:
```

- None

"reactor-http-epoll-2" - Thread t@37

java.lang.Thread.State: RUNNABLE

at io.netty.channel.epoll.Native.epollWait(Native Method)

at io.netty.channel.epoll.Native.epollWait(Native.java:220)

at io.netty.channel.epoll.Native.epollWait(Native.java:213)

at

io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:308)

at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:365)

at

io.netty.util.concurrent.SingleThreadEventExecutor\$4.run(SingleThreadEventExecutor.java:998)

at io.netty.util.internal.ThreadExecutorMap\$2.run(ThreadExecutorMap.java:74)

at

io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)

at java.lang.Thread.run(Thread.java:750)

Locked ownable synchronizers:

- None

"server" - Thread t@36

java.lang.Thread.State: WAITING

at sun.misc.Unsafe.park(Native Method)

- parking to wait for <26a2106e> (a java.util.concurrent.CountDownLatch\$Sync)

at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)

at

java.util.concurrent.locks.AbstractQueuedSynchronizer.parkAndCheckInterrupt(AbstractQueuedSynchronizer.java:836)

at

java.util.concurrent.locks.AbstractQueuedSynchronizer.doAcquireSharedInterruptibly(AbstractQueuedSynchronizer.java:997)

at

java.util.concurrent.locks.AbstractQueuedSynchronizer.acquireSharedInterruptibly(AbstractQueuedSynchronizer.java:1304)

at java.util.concurrent.CountDownLatch.await(CountDownLatch.java:231)

at

reactor.core.publisher.BlockingSingleSubscriber.blockingGet(BlockingSingleSubscriber.java:87)

at reactor.core.publisher.Mono.block(Mono.java:1742)

at

org.springframework.boot.web.embedded.netty.NettyWebServer\$1.run(NettyWebServer.java:180)

Locked ownable synchronizers:

- None

"reactor-http-epoll-1" - Thread t@35

java.lang.Thread.State: RUNNABLE

at io.netty.channel.epoll.Native.epollWait(Native Method)

at io.netty.channel.epoll.Native.epollWait(Native.java:220)

```

    at io.netty.channel.epoll.Native.epollWait(Native.java:213)
    at
io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:308)
    at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:365)
    at
io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.java:998)
    at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)
    at
io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
    at java.lang.Thread.run(Thread.java:750)

```

Locked ownable synchronizers:

- None

```

"boundedElastic-evictor-1" - Thread t@34
  java.lang.Thread.State: TIMED_WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <2c75b0c8> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2078)
    at
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1093)
    at
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:809)
    at java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1074)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1134)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:750)

```

Locked ownable synchronizers:

- None

```

"HikariPool-1 housekeeper" - Thread t@23
  java.lang.Thread.State: TIMED_WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <23c95292> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2078)
    at
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1093)
    at

```

```

java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThread
PoolExecutor.java:809)
    at java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1074)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1134)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:750)

Locked ownable synchronizers:
- None

"/127.0.0.1:33438 to /127.0.0.1:41923 workers Thread 3" - Thread t@12
java.lang.Thread.State: RUNNABLE
    at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
    at sun.nio.ch.EPollArrayWrapper.poll(EPollArrayWrapper.java:269)
    at sun.nio.ch.EPollSelectorImpl.doSelect(EPollSelectorImpl.java:93)
    at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
    - locked <2f43c73a> (a sun.nio.ch.Util$3)
    - locked <19d483dc> (a java.util.Collections$UnmodifiableSet)
    - locked <18a693de> (a sun.nio.ch.EPollSelectorImpl)
    at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:97)
    at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:101)
    at
org.gradle.internal.remote.internal.inet.SocketConnection$SocketInputStream.read(Socket
Connection.java:185)
    at com.esotericsoftware.kryo.io.Input.fill(Input.java:146)
    at com.esotericsoftware.kryo.io.Input.require(Input.java:178)
    at com.esotericsoftware.kryo.io.Input.readByte(Input.java:295)
    at
org.gradle.internal.serialize.kryo.KryoBackedDecoder.readByte(KryoBackedDecoder.java:8
8)
    at
org.gradle.internal.remote.internal.hub.InterHubMessageSerializer$MessageReader.read(I
nterHubMessageSerializer.java:64)
    at
org.gradle.internal.remote.internal.hub.InterHubMessageSerializer$MessageReader.read(I
nterHubMessageSerializer.java:52)
    at
org.gradle.internal.remote.internal.inet.SocketConnection.receive(SocketConnection.jav
a:81)
    at
org.gradle.internal.remote.internal.hub.MessageHub$ConnectionReceive.run(MessageHub.ja
va:270)
    at
org.gradle.internal.concurrent.ExecutorPolicy$CatchAndRecordFailures.onExecute(Executo
rPolicy.java:64)
    at
org.gradle.internal.concurrent.ManagedExecutorImpl$1.run(ManagedExecutorImpl.java:49)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:750)

```

Locked ownable synchronizers:

- Locked <9353778> (a java.util.concurrent.ThreadPoolExecutor\$Worker)

"/127.0.0.1:33438 to /127.0.0.1:41923 workers Thread 2" - Thread t@11

```
java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <71120f9a> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
    at
org.gradle.internal.remote.internal.hub.queue.EndPointQueue.take(EndPointQueue.java:49)
    at
org.gradle.internal.remote.internal.hub.MessageHub$ConnectionDispatch.run(MessageHub.java:322)
    at
org.gradle.internal.concurrent.ExecutorPolicy$CatchAndRecordFailures.onExecute(ExecutorPolicy.java:64)
    at
org.gradle.internal.concurrent.ManagedExecutorImpl$1.run(ManagedExecutorImpl.java:49)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:750)
```

Locked ownable synchronizers:

- Locked <6580cfdd> (a java.util.concurrent.ThreadPoolExecutor\$Worker)

"/127.0.0.1:33438 to /127.0.0.1:41923 workers" - Thread t@10

```
java.lang.Thread.State: WAITING
    at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <40597c95> (a
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
    at java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
    at
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
    at
org.gradle.internal.remote.internal.hub.queue.EndPointQueue.take(EndPointQueue.java:49)
    at
org.gradle.internal.remote.internal.hub.MessageHub$Handler.run(MessageHub.java:403)
    at
org.gradle.internal.concurrent.ExecutorPolicy$CatchAndRecordFailures.onExecute(ExecutorPolicy.java:64)
    at
org.gradle.internal.concurrent.ManagedExecutorImpl$1.run(ManagedExecutorImpl.java:49)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
```

```

    at java.lang.Thread.run(Thread.java:750)

Locked ownable synchronizers:
  - Locked <2ea227af> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"Signal Dispatcher" - Thread t@4
  java.lang.Thread.State: RUNNABLE

  Locked ownable synchronizers:
    - None

"Finalizer" - Thread t@3
  java.lang.Thread.State: WAITING
    at java.lang.Object.wait(Native Method)
    - waiting on <7daecd21> (a java.lang.ref.ReferenceQueue$Lock)
    at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:144)
    at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:165)
    at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:188)

  Locked ownable synchronizers:
    - None

"Reference Handler" - Thread t@2
  java.lang.Thread.State: WAITING
    at java.lang.Object.wait(Native Method)
    - waiting on <51ba8929> (a java.lang.ref.Reference$Lock)
    at java.lang.Object.wait(Object.java:502)
    at java.lang.ref.Reference.tryHandlePending(Reference.java:191)
    at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:153)

  Locked ownable synchronizers:
    - None

"Test worker" - Thread t@1
  java.lang.Thread.State: RUNNABLE
    at sun.management.ThreadImpl.dumpThreads0(Native Method)
    at sun.management.ThreadImpl.dumpAllThreads(ThreadImpl.java:503)
    at sun.management.ThreadImpl.dumpAllThreads(ThreadImpl.java:491)
    at
org.springframework.boot.actuate.management.ThreadDumpEndpoint.getFormattedThreadDump(
ThreadDumpEndpoint.java:51)
    at
org.springframework.boot.actuate.management.ThreadDumpEndpoint.textThreadDump(ThreadDu
mpEndpoint.java:47)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.springframework.util.ReflectionUtils.invokeMethod(ReflectionUtils.java:282)
    at

```

```

org.springframework.boot.actuate.endpoint.invoke.reflect.ReflectiveOperationInvoker.invoke(ReflectiveOperationInvoker.java:74)
    at
org.springframework.boot.actuate.endpoint.annotation.AbstractDiscoveredOperation.invoke(AbstractDiscoveredOperation.java:60)
    at
org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping$ServletWebOperationAdapter.handle(AbstractWebMvcEndpointHandlerMapping.java:357)
    at
org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping$OperationHandler.handle(AbstractWebMvcEndpointHandlerMapping.java:464)
    at sun.reflect.GeneratedMethodAccessor28.invoke(Unknown Source)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at
org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:205)
    at
org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:150)
    at
org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:117)
    at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:903)
    at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:809)
    at
org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87)
    at
org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1072)
    at
org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:965)
    at
org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1006)
    at
org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:898)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:497)
    at
org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)
    at
org.springframework.test.web.servlet.TestDispatcherServlet.service(TestDispatcherServlet.java:72)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:584)

```

```
    at
org.springframework.mock.web.MockFilterChain$ServletFilterProxy.doFilter(MockFilterChain.java:167)
    at org.springframework.mock.web.MockFilterChain.doFilter(MockFilterChain.java:134)
    at org.springframework.test.web.servlet.MockMvc.perform(MockMvc.java:201)
    at
org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.ThreadDumpEndpointDocumentationTests.textThreadDump(ThreadDumpEndpointDocumentationTests.java:182)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at
org.junit.platform.commons.util.ReflectionUtils.invokeMethod(ReflectionUtils.java:725)
    at
org.junit.jupiter.engine.execution.MethodInvocation.proceed(MethodInvocation.java:60)
    at
org.junit.jupiter.engine.execution.InvocationInterceptorChain$ValidatingInvocation.proceed(InvocationInterceptorChain.java:131)
    at
org.junit.jupiter.engine.extension.TimeoutExtension.intercept(TimeoutExtension.java:149)
    at
org.junit.jupiter.engine.extension.TimeoutExtension.interceptTestableMethod(TimeoutExtension.java:140)
    at
org.junit.jupiter.engine.extension.TimeoutExtension.interceptTestMethod(TimeoutExtension.java:84)
    at
org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor$$Lambda$129/1000966072.apply(Unknown Source)
    at
org.junit.jupiter.engine.execution.ExecutableInvoker$ReflectiveInterceptorCall.lambda$ofVoidMethod$0(ExecutableInvoker.java:115)
    at
org.junit.jupiter.engine.execution.ExecutableInvoker$ReflectiveInterceptorCall$$Lambda$130/1912821769.apply(Unknown Source)
    at
org.junit.jupiter.engine.execution.ExecutableInvoker.lambda$invoke$0(ExecutableInvoker.java:105)
    at
org.junit.jupiter.engine.execution.ExecutableInvoker$$Lambda$367/1705277839.apply(Unknown Source)
    at
org.junit.jupiter.engine.execution.InvocationInterceptorChain$InterceptedInvocation.proceed(InvocationInterceptorChain.java:106)
    at
org.junit.jupiter.engine.execution.InvocationInterceptorChain.proceed(InvocationInterceptorChain.java:64)
    at
```

```

org.junit.jupiter.engine.execution.InvocationInterceptorChain.chainAndInvoke(InvocationInter
ceptorChain.java:45)
    at
org.junit.jupiter.engine.execution.InvocationInterceptorChain.invoke(InvocationInterce
ptorChain.java:37)
    at
org.junit.jupiter.engine.execution.ExecutableInvoker.invoke(ExecutableInvoker.java:104
)
    at
org.junit.jupiter.engine.execution.ExecutableInvoker.invoke(ExecutableInvoker.java:98)
    at
org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.lambda$invokeTestMethod$7
(TestMethodTestDescriptor.java:214)
    at
org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor$$Lambda$814/535910992.exe
cute(Unknown Source)
    at
org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCol
lector.java:73)
    at
org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.invokeTestMethod(TestMeth
odTestDescriptor.java:210)
    at
org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.execute(TestMethodTestDes
criptor.java:135)
    at
org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.execute(TestMethodTestDes
criptor.java:66)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$
6(NodeTestTask.java:151)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$229/2103569237.exe
cute(Unknown Source)
    at
org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCol
lector.java:73)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$
8(NodeTestTask.java:141)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$228/873634936.invo
ke(Unknown Source)
    at org.junit.platform.engine.support.hierarchical.Node.around(Node.java:137)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$
9(NodeTestTask.java:139)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$227/951031848.exe
cute(Unknown Source)
    at

```

```

org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.executeRecursively(NodeTestTask.java:138)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.execute(NodeTestTask.java:95)
    at
org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService$$$Lambda$233/1670313965.accept(Unknown Source)
    at java.util.ArrayList.forEach(ArrayList.java:1259)
    at
org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService.invokeAll(SameThreadHierarchicalTestExecutorService.java:41)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$6(NodeTestTask.java:155)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask$$$Lambda$229/2103569237.execute(Unknown Source)
    at
org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$8(NodeTestTask.java:141)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask$$$Lambda$228/873634936.invoke(Unknown Source)
    at org.junit.platform.engine.support.hierarchical.Node.around(Node.java:137)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$9(NodeTestTask.java:139)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask$$$Lambda$227/951031848.execute(Unknown Source)
    at
org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.executeRecursively(NodeTestTask.java:138)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.execute(NodeTestTask.java:95)
    at
org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService$$$Lambda$233/1670313965.accept(Unknown Source)
    at java.util.ArrayList.forEach(ArrayList.java:1259)
    at

```

```
org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService.invokeAll(SameThreadHierarchicalTestExecutorService.java:41)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$6(NodeTestTask.java:155)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$229/2103569237.execute(Unknown Source)
    at
org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$8(NodeTestTask.java:141)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$228/873634936.invoke(Unknown Source)
    at org.junit.platform.engine.support.hierarchical.Node.around(Node.java:137)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$9(NodeTestTask.java:139)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$227/951031848.execute(Unknown Source)
    at
org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.executeRecursively(NodeTestTask.java:138)
    at
org.junit.platform.engine.support.hierarchical.NodeTestTask.execute(NodeTestTask.java:95)
    at
org.junit.platform.engine.support.hierarchical.SameThreadHierarchicalTestExecutorService.submit(SameThreadHierarchicalTestExecutorService.java:35)
    at
org.junit.platform.engine.support.hierarchical.HierarchicalTestExecutor.execute(HierarchicalTestExecutor.java:57)
    at
org.junit.platform.engine.support.hierarchical.HierarchicalTestEngine.execute(HierarchicalTestEngine.java:54)
    at
org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:107)
    at
org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:88)
    at
org.junit.platform.launcher.core.EngineExecutionOrchestrator.lambda$execute$0(EngineExecutionOrchestrator.java:54)
```

```

    at
org.junit.platform.launcher.core.EngineExecutionOrchestrator$$Lambda$179/361268035.accept(Unknown Source)
    at
org.junit.platform.launcher.core.EngineExecutionOrchestrator.withInterceptedStreams(EngineExecutionOrchestrator.java:67)
    at
org.junit.platform.launcher.core.EngineExecutionOrchestrator.execute(EngineExecutionOrchestrator.java:52)
    at
org.junit.platform.launcher.core.DefaultLauncher.execute(DefaultLauncher.java:114)
    at
org.junit.platform.launcher.core.DefaultLauncher.execute(DefaultLauncher.java:86)
    at
org.junit.platform.launcher.core.DefaultLauncherSession$DelegatingLauncher.execute(DefaultLauncherSession.java:86)
    at
org.junit.platform.launcher.core.SessionPerRequestLauncher.execute(SessionPerRequestLauncher.java:53)
    at
org.gradle.api.internal.tasks.testing.junitplatform.JUnitPlatformTestClassProcessor$CollectAllTestClassesExecutor.processAllTestClasses(JUnitPlatformTestClassProcessor.java:99)
    at
org.gradle.api.internal.tasks.testing.junitplatform.JUnitPlatformTestClassProcessor$CollectAllTestClassesExecutor.access$000(JUnitPlatformTestClassProcessor.java:79)
    at
org.gradle.api.internal.tasks.testing.junitplatform.JUnitPlatformTestClassProcessor.stop(JUnitPlatformTestClassProcessor.java:75)
    at
org.gradle.api.internal.tasks.testing.SuiteTestClassProcessor.stop(SuiteTestClassProcessor.java:62)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at
org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:36)
    at
org.gradle.internal.dispatch.ReflectionDispatch.dispatch(ReflectionDispatch.java:24)
    at
org.gradle.internal.dispatch.ContextClassLoaderDispatch.dispatch(ContextClassLoaderDispatch.java:33)
    at
org.gradle.internal.dispatch.ProxyDispatchAdapter$DispatchingInvocationHandler.invoke(ProxyDispatchAdapter.java:94)
    at com.sun.proxy.$Proxy2.stop(Unknown Source)
    at
org.gradle.api.internal.tasks.testing.worker.TestWorker$3.run(TestWorker.java:193)
    at

```

```
org.gradle.api.internal.tasks.testing.worker.TestWorker.executeAndMaintainThreadName(TestWorker.java:129)
    at
org.gradle.api.internal.tasks.testing.worker.TestWorker.execute(TestWorker.java:100)
    at
org.gradle.api.internal.tasks.testing.worker.TestWorker.execute(TestWorker.java:60)
    at
org.gradle.process.internal.worker.child.ActionExecutionWorker.execute(ActionExecutionWorker.java:56)
    at
org.gradle.process.internal.worker.child.SystemApplicationClassLoaderWorker.call(SystemApplicationClassLoaderWorker.java:113)
    at
org.gradle.process.internal.worker.child.SystemApplicationClassLoaderWorker.call(SystemApplicationClassLoaderWorker.java:65)
    at
worker.org.gradle.process.internal.worker.GradleWorkerMain.run(GradleWorkerMain.java:69)
    at
worker.org.gradle.process.internal.worker.GradleWorkerMain.main(GradleWorkerMain.java:74)
```

Locked ownable synchronizers:

- None