Table of Contents

1. Overview ................................................................. 2
   1.1. URLs ............................................................. 2
   1.2. Timestamps ....................................................... 2

2. Audit Events (**auditevents**) .................................. 3
   2.1. Retrieving Audit Events .................................... 3
       2.1.1. Query Parameters ...................................... 3
       2.1.2. Response Structure ..................................... 3

3. Beans (**beans**) ..................................................... 5
   3.1. Retrieving the Beans ......................................... 5
       3.1.1. Response Structure ..................................... 6

4. Caches (**caches**) .................................................. 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 (**conditions**) ............ 11
   5.1. Retrieving the Report ...................................... 11
       5.1.1. Response Structure ..................................... 11

6. Configuration Properties (**configprops**) .................. 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 (**env**) ............................................... 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 (**flyway**) .................................................. 24
   8.1. Retrieving the Migrations .................................. 24
       8.1.1. Response Structure ..................................... 24

9. Health (**health**) .................................................. 26
   9.1. Retrieving the Health of the Application ............. 26
       9.1.1. Response Structure ..................................... 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.2.1. Response Structure</td>
<td>82</td>
</tr>
<tr>
<td>22.3. Deleting a Session</td>
<td>82</td>
</tr>
<tr>
<td>23. Shutdown (shutdown)</td>
<td>83</td>
</tr>
<tr>
<td>23.1. Shutting Down the Application</td>
<td>83</td>
</tr>
<tr>
<td>23.1.1. Response Structure</td>
<td>83</td>
</tr>
<tr>
<td>24. Application Startup (startup)</td>
<td>84</td>
</tr>
<tr>
<td>24.1. Retrieving the Application Startup Steps</td>
<td>84</td>
</tr>
<tr>
<td>24.1.1. Retrieving a snapshot of the Application Startup Steps</td>
<td>84</td>
</tr>
<tr>
<td>24.1.2. Draining the Application Startup Steps</td>
<td>85</td>
</tr>
<tr>
<td>24.1.3. Response Structure</td>
<td>86</td>
</tr>
<tr>
<td>25. Thread Dump (threaddump)</td>
<td>88</td>
</tr>
<tr>
<td>25.1. Retrieving the Thread Dump as JSON</td>
<td>88</td>
</tr>
<tr>
<td>25.1.1. Response Structure</td>
<td>92</td>
</tr>
<tr>
<td>25.2. Retrieving the Thread Dump as Text</td>
<td>94</td>
</tr>
</tbody>
</table>
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:

```bash
```

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:

```json
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 121

{
   "events" : [ {
      "timestamp" : "2021-07-22T17:48:30.462Z",
      "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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>after</code></td>
<td>Restricts the events to those that occurred after the given time. Optional.</td>
</tr>
<tr>
<td><code>principal</code></td>
<td>Restricts the events to those with the given principal. Optional.</td>
</tr>
<tr>
<td><code>type</code></td>
<td>Restricts the events to those with the given type. Optional.</td>
</tr>
</tbody>
</table>

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:
<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>events</td>
<td>Array</td>
<td>An array of audit events.</td>
</tr>
<tr>
<td>events[].timestamp</td>
<td>String</td>
<td>The timestamp of when the event occurred.</td>
</tr>
<tr>
<td>events[].principal</td>
<td>String</td>
<td>The principal that triggered the event.</td>
</tr>
<tr>
<td>events[].type</td>
<td>String</td>
<td>The type of the event.</td>
</tr>
</tbody>
</table>
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:
3.1.1. Response Structure

The response contains details of the application’s beans. The following table describes the structure of the response:
<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contexts</td>
<td>Object</td>
<td>Application contexts keyed by id.</td>
</tr>
<tr>
<td>contexts.*.parentId</td>
<td>String</td>
<td>Id of the parent application context, if any.</td>
</tr>
<tr>
<td>contexts.*.beans</td>
<td>Object</td>
<td>Beans in the application context keyed by name.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.aliases</td>
<td>Array</td>
<td>Names of any aliases.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.scope</td>
<td>String</td>
<td>Scope of the bean.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.type</td>
<td>String</td>
<td>Fully qualified type of the bean.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.resource</td>
<td>String</td>
<td>Resource in which the bean was defined, if any.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.dependencies</td>
<td>Array</td>
<td>Names of any dependencies.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacheManagers</td>
<td>Object</td>
<td>Cache managers keyed by id.</td>
</tr>
</tbody>
</table>
### 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cacheManager</td>
<td>Name of the cacheManager to qualify the cache. May be omitted if the cache name is unique.</td>
</tr>
</tbody>
</table>

#### 4.2.2. Response Structure

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

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Cache name.</td>
</tr>
<tr>
<td>cacheManager</td>
<td>String</td>
<td>Cache manager name.</td>
</tr>
</tbody>
</table>
### 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cacheManager</code></td>
<td>Name of the cacheManager to qualify the cache. May be omitted if the cache name is unique.</td>
</tr>
</tbody>
</table>
Chapter 5. Conditions Evaluation Report

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: 3255

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

The response contains details of the application's condition evaluation. The following table describes the structure of the response:
<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contexts</td>
<td>Object</td>
<td>Application contexts keyed by id.</td>
</tr>
<tr>
<td>contexts.*.positiveMatches</td>
<td>Object</td>
<td>Classes and methods with conditions that were matched.</td>
</tr>
<tr>
<td>contexts.<em>.positiveMatches.</em>.condition</td>
<td>String</td>
<td>Name of the condition.</td>
</tr>
<tr>
<td>contexts.<em>.positiveMatches.</em>.message</td>
<td>String</td>
<td>Details of why the condition was matched.</td>
</tr>
<tr>
<td>contexts.*.negativeMatches</td>
<td>Object</td>
<td>Classes and methods with conditions that were not matched.</td>
</tr>
<tr>
<td>contexts.<em>.negativeMatches.</em>.notMatched</td>
<td>Array</td>
<td>Conditions that were matched.</td>
</tr>
<tr>
<td>contexts.<em>.negativeMatches.</em>.notMatched.*.condition</td>
<td>String</td>
<td>Name of the condition.</td>
</tr>
<tr>
<td>contexts.<em>.negativeMatches.</em>.notMatched.*.message</td>
<td>String</td>
<td>Details of why the condition was not matched.</td>
</tr>
<tr>
<td>contexts.<em>.negativeMatches.</em>.matched</td>
<td>Array</td>
<td>Conditions that were matched.</td>
</tr>
<tr>
<td>contexts.<em>.negativeMatches.</em>.matched.*.condition</td>
<td>String</td>
<td>Name of the condition.</td>
</tr>
<tr>
<td>contexts.<em>.negativeMatches.</em>.matched.*.message</td>
<td>String</td>
<td>Details of why the condition was matched.</td>
</tr>
<tr>
<td>contexts.*.unconditionalClasses</td>
<td>Array</td>
<td>Names of unconditional auto-configuration classes if any.</td>
</tr>
<tr>
<td>contexts.*.parentId</td>
<td>String</td>
<td>Id of the parent application context, if any.</td>
</tr>
</tbody>
</table>
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:

```bash
$ 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": ":value",
      "origin": ":origin 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": { },
}
6.1.1. Response Structure

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

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contexts</td>
<td>Object</td>
<td>Application contexts keyed by id.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em></td>
<td>Object</td>
<td><code>@ConfigurationProperties</code> beans keyed by bean name.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.prefix</td>
<td>String</td>
<td>Prefix applied to the names of the bean's properties.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.properties</td>
<td>Object</td>
<td>Properties of the bean as name-value pairs.</td>
</tr>
</tbody>
</table>
### 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:

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

The resulting response is similar to the following:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contexts.<em>.beans.</em>.inputs</td>
<td>Object</td>
<td>Origin and value of the configuration property used when binding to this bean.</td>
</tr>
<tr>
<td>contexts.*.parentId</td>
<td>String</td>
<td>Id of the parent application context, if any.</td>
</tr>
</tbody>
</table>
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-\norg.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": {
                        }
                    }
                }
            }
        }
    }
}

18
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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contexts</td>
<td>Object</td>
<td>Application contexts keyed by id.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em></td>
<td>Object</td>
<td><code>@ConfigurationProperties</code> beans keyed by bean name.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.prefix</td>
<td>String</td>
<td>Prefix applied to the names of the bean's properties.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.properties</td>
<td>Object</td>
<td>Properties of the bean as name-value pairs.</td>
</tr>
<tr>
<td>contexts.<em>.beans.</em>.inputs</td>
<td>Object</td>
<td>Origin and value of the configuration property used when binding to this bean.</td>
</tr>
<tr>
<td>contexts.*.parentId</td>
<td>String</td>
<td>Id of the parent application context, if any.</td>
</tr>
</tbody>
</table>
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: 825

```json
{
    "activeProfiles": [ ],
    "propertySources": [ {
        "name": "systemProperties",
        "properties": {
            "java.runtime.name": {
                "value": "OpenJDK Runtime Environment"
            },
            "java.vm.version": {
                "value": "25.292-b10"
            },
            "java.vm.vendor": {
                "value": "AdoptOpenJDK"
            }
        }
    }, {
        "name": "systemEnvironment",
        "properties": {
            "JAVA_HOME": {
                "value": "/opt/openjdk",
                "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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activeProfiles</td>
<td>Array</td>
<td>Names of the active profiles, if any.</td>
</tr>
<tr>
<td>propertySources</td>
<td>Array</td>
<td>Property sources in order of precedence.</td>
</tr>
</tbody>
</table>
### 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:

```bash
$ 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: 517

{
   "property" : {
      "source" : "Config resource 'class path resource [application.properties]' via location 'classpath:/'",
      "value" : "1000"
   },
   "activeProfiles" : [ ],
   "propertySources" : [ {
      "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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>Object</td>
<td>Property from the environment, if found.</td>
</tr>
<tr>
<td>property.source</td>
<td>String</td>
<td>Name of the source of the property.</td>
</tr>
<tr>
<td>property.value</td>
<td>String</td>
<td>Value of the property.</td>
</tr>
<tr>
<td>activeProfiles</td>
<td>Array</td>
<td>Names of the active profiles, if any.</td>
</tr>
<tr>
<td>propertySources</td>
<td>Array</td>
<td>Property sources in order of precedence.</td>
</tr>
<tr>
<td>propertySources[].name</td>
<td>String</td>
<td>Name of the property source.</td>
</tr>
<tr>
<td>propertySources[].property</td>
<td>Object</td>
<td>Property in the property source, if any.</td>
</tr>
<tr>
<td>propertySources[].property.value</td>
<td>Varies</td>
<td>Value of the property.</td>
</tr>
<tr>
<td>propertySources[].property.origin</td>
<td>String</td>
<td>Origin of the property, if any.</td>
</tr>
</tbody>
</table>
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:

```bash
$ 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: 516

{
   "contexts": {
      "application": {
         "flywayBeans": {
            "flyway": {
               "migrations": [ {
                  "type": "SQL",
                  "checksum": -156244537,
                  "version": "1",
                  "description": "init",
                  "script": "V1__init.sql",
                  "state": "SUCCESS",
                  "installedBy": "SA",
                  "installedOn": "2021-07-22T17:48:29.433Z",
                  "installedRank": 1,
                  "executionTime": 18
               } ]
            }
         }
      }
   }
}
```

8.1.1. Response Structure

The response contains details of the application’s Flyway migrations. The following table describes the structure of the response:
<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contexts</td>
<td>Object</td>
<td>Application contexts keyed by id</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations</td>
<td>Array</td>
<td>Migrations performed by the Flyway instance, keyed by Flyway bean name.</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].checksum</td>
<td>Number</td>
<td>Checksum of the migration, if any.</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].description</td>
<td>String</td>
<td>Description of the migration, if any.</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].executionTime</td>
<td>Number</td>
<td>Execution time in milliseconds of an applied migration.</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].installedBy</td>
<td>String</td>
<td>User that installed the applied migration, if any.</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].installedOn</td>
<td>String</td>
<td>Timestamp of when the applied migration was installed, if any.</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].installedRank</td>
<td>Number</td>
<td>Rank of the applied migration, if any. Later migrations have higher ranks.</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].script</td>
<td>String</td>
<td>Name of the script used to execute the migration, if any.</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].state</td>
<td>String</td>
<td>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)</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].type</td>
<td>String</td>
<td>Type of the migration. (SCHEMA, BASELINE, DELETE, SQL, UNDO_SQL, JDBC, UNDO_JDBC, SPRING_JDBC, UNDO_SPRING_JDBC, CUSTOM, UNDO_CUSTOM)</td>
</tr>
<tr>
<td>contexts.<em>.flywayBeans.</em>.migrations[].version</td>
<td>String</td>
<td>Version of the database after applying the migration, if any.</td>
</tr>
<tr>
<td>contexts.*.parentId</td>
<td>String</td>
<td>Id of the parent application context, if any.</td>
</tr>
</tbody>
</table>
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:

```bash
$ 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": 325496897536,
                "free": 269835796480,
                "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:
<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>String</td>
<td>Overall status of the application.</td>
</tr>
<tr>
<td>components</td>
<td>Object</td>
<td>The components that make up the health.</td>
</tr>
<tr>
<td>components.*.status</td>
<td>String</td>
<td>Status of a specific part of the application.</td>
</tr>
<tr>
<td>components.*.components</td>
<td>Object</td>
<td>The nested components that make up the health.</td>
</tr>
<tr>
<td>components.*.details</td>
<td>Object</td>
<td>Details of the health of a specific part of the application. Presence is controlled by management.endpoint.health.show-details.</td>
</tr>
</tbody>
</table>

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:

```bash
$ 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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>String</td>
<td>Status of a specific part of the application</td>
</tr>
<tr>
<td>details</td>
<td>Object</td>
<td>Details of the health of a specific part of the application</td>
</tr>
</tbody>
</table>
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 `{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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>String</td>
<td>Status of a specific part of the application</td>
</tr>
<tr>
<td>details</td>
<td>Object</td>
<td>Details of the health of a specific part of the application.</td>
</tr>
</tbody>
</table>
Chapter 10. Heap Dump (heapdump)

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

10.1. Retrieving the Heap Dump

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

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

The preceding example results in a file named heapdump being written to the current working directory.
Chapter 11. HTTP Trace (**httptrace**)

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

### 11.1. Retrieving the Traces

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

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

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Content-Length: 503

{
  "traces" : [
    {
      "principal" : {
        "name" : "alice"
      },
      "session" : {
        "id" : "1cd6e481-2ae3-4d9f-97c2-5ab4c940fb6a"
      },
      "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:
<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>traces</td>
<td>Array</td>
<td>An array of traced HTTP request-response exchanges.</td>
</tr>
<tr>
<td>traces[].timestamp</td>
<td>String</td>
<td>Timestamp of when the traced exchange occurred.</td>
</tr>
<tr>
<td>traces[].principal</td>
<td>Object</td>
<td>Principal of the exchange, if any.</td>
</tr>
<tr>
<td>traces[].principal.name</td>
<td>String</td>
<td>Name of the principal.</td>
</tr>
<tr>
<td>traces[].request.method</td>
<td>String</td>
<td>HTTP method of the request.</td>
</tr>
<tr>
<td>traces[].request.remoteAddres</td>
<td>String</td>
<td>Remote address from which the request was received, if known.</td>
</tr>
<tr>
<td>traces[].request.uri</td>
<td>String</td>
<td>URI of the request.</td>
</tr>
<tr>
<td>traces[].request.headers</td>
<td>Object</td>
<td>Headers of the request, keyed by header name.</td>
</tr>
<tr>
<td>traces[].request.headers.*.[]</td>
<td>Array</td>
<td>Values of the header</td>
</tr>
<tr>
<td>traces[].response.status</td>
<td>Number</td>
<td>Status of the response</td>
</tr>
<tr>
<td>traces[].response.headers</td>
<td>Object</td>
<td>Headers of the response, keyed by header name.</td>
</tr>
<tr>
<td>traces[].response.headers.*.[ ]</td>
<td>Array</td>
<td>Values of the header</td>
</tr>
<tr>
<td>traces[].session</td>
<td>Object</td>
<td>Session associated with the exchange, if any.</td>
</tr>
<tr>
<td>traces[].session.id</td>
<td>String</td>
<td>ID of the session.</td>
</tr>
<tr>
<td>traces[].timeTaken</td>
<td>Number</td>
<td>Time, in milliseconds, taken to handle the exchange.</td>
</tr>
</tbody>
</table>
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: 233

{
    "git" : {
        "commit" : {
            "time" : "+53526-10-27T20:00:34Z",
            "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 build and git contributions.

Build Response Structure

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

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>artifact</td>
<td>String</td>
<td>Artifact ID of the application, if any.</td>
</tr>
<tr>
<td>group</td>
<td>String</td>
<td>Group ID of the application, if any.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the application, if any.</td>
</tr>
<tr>
<td>Path</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>version</td>
<td>String</td>
<td>Version of the application, if any.</td>
</tr>
<tr>
<td>time</td>
<td>Varies</td>
<td>Timestamp of when the application was built, if any.</td>
</tr>
</tbody>
</table>

**Git Response Structure**

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

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>branch</td>
<td>String</td>
<td>Name of the Git branch, if any.</td>
</tr>
<tr>
<td>commit</td>
<td>Object</td>
<td>Details of the Git commit, if any.</td>
</tr>
<tr>
<td>commit.time</td>
<td>Varies</td>
<td>Timestamp of the commit, if any.</td>
</tr>
<tr>
<td>commit.id</td>
<td>String</td>
<td>ID of the commit, if any.</td>
</tr>
</tbody>
</table>
Chapter 13. Spring Integration graph

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: 961

{
    "contentDescriptor": {
        "providerVersion": "5.5.2",
        "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:

```bash
$ 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" : [ ],
            "deploymentId" : "6976097843",
            "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:
<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contexts</td>
<td>Object</td>
<td>Application contexts keyed by id</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets</td>
<td>Array</td>
<td>Change sets made by the Liquibase beans, keyed by bean name.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].author</td>
<td>String</td>
<td>Author of the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].changeLog</td>
<td>String</td>
<td>Change log that contains the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].comments</td>
<td>String</td>
<td>Comments on the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].contexts</td>
<td>Array</td>
<td>Contexts of the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].dateExecuted</td>
<td>String</td>
<td>Timestamp of when the change set was executed.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].deploymentId</td>
<td>String</td>
<td>ID of the deployment that ran the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].description</td>
<td>String</td>
<td>Description of the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].execType</td>
<td>String</td>
<td>Execution type of the change set (EXECUTED, FAILED, SKIPPED, RERAN, MARK_RAN)</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].id</td>
<td>String</td>
<td>ID of the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].labels</td>
<td>Array</td>
<td>Labels associated with the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].checksum</td>
<td>String</td>
<td>Checksum of the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].orderExecuted</td>
<td>Number</td>
<td>Order of the execution of the change set.</td>
</tr>
<tr>
<td>contexts.<em>.liquibaseBeans.</em>.changeSets[].tag</td>
<td>String</td>
<td>Tag associated with the change set, if any.</td>
</tr>
<tr>
<td>contexts.*.parentId</td>
<td>String</td>
<td>Id of the parent application context, if any.</td>
</tr>
</tbody>
</table>
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:

```bash
$ 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

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.AnnotationConfigServletWebServerApplicationContext@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]
2017-08-08 17:12:31.978 INFO 19866 --- [ost-startStop-1]
o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
```

40
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

...
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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>levels</td>
<td>Array</td>
<td>Levels support by the logging system.</td>
</tr>
<tr>
<td>loggers</td>
<td>Object</td>
<td>Loggers keyed by name.</td>
</tr>
<tr>
<td>groups</td>
<td>Object</td>
<td>Logger groups keyed by name</td>
</tr>
<tr>
<td>loggers.*.configuredLevel</td>
<td>String</td>
<td>Configured level of the logger, if any.</td>
</tr>
<tr>
<td>loggers.*.effectiveLevel</td>
<td>String</td>
<td>Effective level of the logger.</td>
</tr>
<tr>
<td>groups.*.configuredLevel</td>
<td>String</td>
<td>Configured level of the logger group, if any.</td>
</tr>
<tr>
<td>groups.*.members</td>
<td>Array</td>
<td>Loggers that are part of this group</td>
</tr>
</tbody>
</table>

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:

```bash
$ 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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuredLevel</td>
<td>String</td>
<td>Configured level of the logger, if any.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuredLevel</td>
<td>String</td>
<td>Configured level of the logger group, if any.</td>
</tr>
<tr>
<td>members</td>
<td>Array</td>
<td>Loggers that are part of this group</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuredLevel</td>
<td>String</td>
<td>Level for the logger. May be omitted to clear the level.</td>
</tr>
</tbody>
</table>

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:

```bash
$ 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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuredLevel</td>
<td>String</td>
<td>Level for the logger. May be omitted to clear the level.</td>
</tr>
</tbody>
</table>

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:

```bash
$ 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

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:3973/actuator/mappings' -i -X GET
```

The resulting response is similar to the following:

```
HTTP/1.1 200 OK
Content-Type: application/vnd.spring-boot.actuator.v3+json
Transfer-Encoding: chunked
Date: Thu, 22 Jul 2021 17:48:13 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": {
                "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
                } ],
```
"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.WebMvcLinksHandler",
    "name" : "links",
    "descriptor" : "(Ljavax/servlet/http/HttpServletRequest;Ljavax/servlet/http/HttpServletResponse;)Ljava/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.MappingsEndpointServletDocumentationTests$ExampleController#example()",
  "predicate" : "{POST [/], params [a!=alpha], headers [X-Custom=Foo], consumes [application/json || !application/xml], produces [text/plain]}",
  "details" : {
    "handlerMethod" : {
      "className" : "org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.MappingsEndpointServletDocumentationTests$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.MappingsEndpointServletDocumentationTests$ExampleController#example()",
          "predicate" : "{POST [/], params [a!=alpha], headers [X-Custom=Foo], consumes [application/json || !application/xml], produces [text/plain]}",
          "details" : {
            "handlerMethod" : {
              "className" : "org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.MappingsEndpointServletDocumentationTests$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.MappingsEndpointServletDocumentationTests$ExampleController#example()",
                  "predicate" : "{POST [/], params [a!=alpha], headers [X-Custom=Foo], consumes [application/json || !application/xml], produces [text/plain]}",
                  "details" : {
                    "handlerMethod" : {
                      "className" : "org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.MappingsEndpointServletDocumentationTests$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.MappingsEndpointServletDocumentationTests$ExampleController#example()",
                          "predicate" : "{POST [/], params [a!=alpha], headers [X-Custom=Foo], consumes [application/json || !application/xml], produces [text/plain]}",
                          "details" : {
                            "handlerMethod" : {
                              "className" : "org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.MappingsEndpointServletDocumentationTests$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.MappingsEndpointServletDocumentationTests$ExampleController#example()",
                                  "predicate" : "{POST [/], params [a!=alpha], headers [X-Custom=Foo], consumes [application/json || !application/xml], produces [text/plain]}",
                                  "details" : {
                                    "handlerMethod" : {
                                      "className" : "org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.MappingsEndpointServletDocumentationTests$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.MappingsEndpointServletDocumentationTests$ExampleController#example()",
                                          "predicate" : "{POST [/], params [a!=alpha], headers [X-Custom=Foo], consumes [application/json || !application/xml], produces [text/plain]}",
                                          "details" : {
                                            "handlerMethod" : {
                                              "className" : "org.springframework.boot.actuate.autoconfigure.endpoint.web.documentation.MappingsEndpointServletDocumentationTests$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.MappingsEndpointServletDocumentationTests$ExampleController#example()",
                                                  "predicate" : "{POST /[
"}
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.OrderedRequestContextFilter"
    }]
}
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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>contexts</td>
<td>Object</td>
<td>Application contexts keyed by id.</td>
</tr>
<tr>
<td>contexts.*.mappings</td>
<td>Object</td>
<td>Mappings in the context, keyed by mapping type.</td>
</tr>
<tr>
<td>contexts.*.mappings.dispatcherServlets</td>
<td>Object</td>
<td>Dispatcher servlet mappings, if any.</td>
</tr>
<tr>
<td>contexts.*.mappings.servletFilters</td>
<td>Array</td>
<td>Servlet filter mappings, if any.</td>
</tr>
<tr>
<td>contexts.*.mappings.servlets</td>
<td>Array</td>
<td>Servlet mappings, if any.</td>
</tr>
<tr>
<td>contexts.*.mappings.dispatcherHandlers</td>
<td>Object</td>
<td>Dispatcher handler mappings, if any.</td>
</tr>
<tr>
<td>contexts.*.parentId</td>
<td>String</td>
<td>Id of the parent application context, if any.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Array</td>
<td>Dispatcher servlet mappings, if any, keyed by dispatcher servlet bean name.</td>
</tr>
<tr>
<td>Path</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>*.[].details</td>
<td>Object</td>
<td>Additional implementation-specific details about the mapping. Optional.</td>
</tr>
<tr>
<td>*.[].handler</td>
<td>String</td>
<td>Handler for the mapping.</td>
</tr>
<tr>
<td>*.[].predicate</td>
<td>String</td>
<td>Predicate for the mapping.</td>
</tr>
<tr>
<td>*.[].details.handlerMethod</td>
<td>Object</td>
<td>Details of the method, if any, that will handle requests to this mapping.</td>
</tr>
<tr>
<td>*.[].details.handlerMethod.className</td>
<td>Varies</td>
<td>Fully qualified name of the class of the method.</td>
</tr>
<tr>
<td>*.[].details.handlerMethod.name</td>
<td>Varies</td>
<td>Name of the method.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions</td>
<td>Object</td>
<td>Details of the request mapping conditions.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.consumes</td>
<td>Varies</td>
<td>Details of the consumes condition.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.consumes.[].mediaType</td>
<td>Varies</td>
<td>Consumed media type.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.consumes.[].negated</td>
<td>Varies</td>
<td>Whether the media type is negated.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.headers</td>
<td>Varies</td>
<td>Details of the headers condition.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.headers.[].name</td>
<td>Varies</td>
<td>Name of the header.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.headers.[].value</td>
<td>Varies</td>
<td>Required value of the header, if any.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.headers.[].negated</td>
<td>Varies</td>
<td>Whether the value is negated.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.methods</td>
<td>Varies</td>
<td>HTTP methods that are handled.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.params</td>
<td>Varies</td>
<td>Details of the params condition.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.params.[].name</td>
<td>Varies</td>
<td>Name of the parameter.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.params.[].value</td>
<td>Varies</td>
<td>Required value of the parameter, if any.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.params.[].negated</td>
<td>Varies</td>
<td>Whether the value is negated.</td>
</tr>
</tbody>
</table>
### 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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*._mappings</td>
<td>Array</td>
<td>Mappings of the servlet.</td>
</tr>
<tr>
<td>*.name</td>
<td>String</td>
<td>Name of the servlet.</td>
</tr>
<tr>
<td>*.className</td>
<td>String</td>
<td>Class name of the servlet.</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.servletNameMappings</td>
<td>Array</td>
<td>Names of the servlets to which the filter is mapped.</td>
</tr>
<tr>
<td>*.urlPatternMappings</td>
<td>Array</td>
<td>URL pattern to which the filter is mapped.</td>
</tr>
<tr>
<td>*.name</td>
<td>String</td>
<td>Name of the filter.</td>
</tr>
<tr>
<td>*.className</td>
<td>String</td>
<td>Class name of the filter.</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.details.requestMappingConditions.patterns</td>
<td>Varies</td>
<td>Patterns identifying the paths handled by the mapping.</td>
</tr>
<tr>
<td>*.details.requestMappingConditions.produces</td>
<td>Varies</td>
<td>Details of the produces condition.</td>
</tr>
<tr>
<td>*.details.requestMappingConditions.produces.[].mediatype</td>
<td>Varies</td>
<td>Produced media type.</td>
</tr>
<tr>
<td>*.details.requestMappingConditions.produces.[].negated</td>
<td>Varies</td>
<td>Whether the media type is negated.</td>
</tr>
<tr>
<td>Path</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>*</td>
<td>Array</td>
<td>Dispatcher handler mappings, if any, keyed by dispatcher handler bean name.</td>
</tr>
<tr>
<td>*[].details</td>
<td>Object</td>
<td>Additional implementation-specific details about the mapping. Optional.</td>
</tr>
<tr>
<td>*[].handler</td>
<td>String</td>
<td>Handler for the mapping.</td>
</tr>
<tr>
<td>*[].predicate</td>
<td>String</td>
<td>Predicate for the mapping.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions</td>
<td>Object</td>
<td>Details of the request mapping conditions.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.consumes</td>
<td>Array</td>
<td>Details of the consumes condition</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.consumes[].mediaType</td>
<td>String</td>
<td>Consumed media type.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.consumes[].negated</td>
<td>Boolean</td>
<td>Whether the media type is negated.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.headers</td>
<td>Array</td>
<td>Details of the headers condition.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.headers[].name</td>
<td>String</td>
<td>Name of the header.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.headers[].value</td>
<td>String</td>
<td>Required value of the header, if any.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.headers[].negated</td>
<td>Boolean</td>
<td>Whether the value is negated.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.methods</td>
<td>Array</td>
<td>HTTP methods that are handled.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.params</td>
<td>Array</td>
<td>Details of the params condition.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.params[].name</td>
<td>String</td>
<td>Name of the parameter.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.params[].value</td>
<td>String</td>
<td>Required value of the parameter, if any.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.params[].negated</td>
<td>Boolean</td>
<td>Whether the value is negated.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.patterns</td>
<td>Array</td>
<td>Patterns identifying the paths handled by the mapping.</td>
</tr>
<tr>
<td>*[].details.requestMappingConditions.produces</td>
<td>Array</td>
<td>Details of the produces condition.</td>
</tr>
<tr>
<td>Path</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------------------------------</td>
<td>--------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.produces.[].mediaType</td>
<td>String</td>
<td>Produced media type.</td>
</tr>
<tr>
<td>*.[].details.requestMappingConditions.produces.[].negated</td>
<td>Boolean</td>
<td>Whether the media type is negated.</td>
</tr>
<tr>
<td>*.[].details.handlerMethod</td>
<td>Object</td>
<td>Details of the method, if any, that will handle requests to this mapping.</td>
</tr>
<tr>
<td>*.[].details.handlerMethod.className</td>
<td>String</td>
<td>Fully qualified name of the class of the method.</td>
</tr>
<tr>
<td>*.[].details.handlerMethod.name</td>
<td>String</td>
<td>Name of the method.</td>
</tr>
<tr>
<td>*.[].details.handlerFunction</td>
<td>Object</td>
<td>Details of the function, if any, that will handle requests to this mapping.</td>
</tr>
<tr>
<td>*.[].details.handlerFunction.className</td>
<td>String</td>
<td>Fully qualified name of the class of the function.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>names</td>
<td>Array</td>
<td>Names of the known metrics.</td>
</tr>
</tbody>
</table>

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:
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag</td>
<td>A tag to use for drill-down in the form name:value.</td>
</tr>
</tbody>
</table>

18.2.2. Response structure

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

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the metric</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the metric</td>
</tr>
<tr>
<td>baseUnit</td>
<td>String</td>
<td>Base unit of the metric</td>
</tr>
<tr>
<td>measurements</td>
<td>Array</td>
<td>Measurements of the metric</td>
</tr>
</tbody>
</table>
18.3. Drilling Down

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

```
$ curl 'http://localhost:8080/actuator/metrics/jvm.memory.max?tag=area%3Anonheap&tag=id%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: 2374

# 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"}, 7.1827456E7
jvm_memory_max_bytes{area="heap",id="PS Old Gen"}, 7.1617408E8
jvm_memory_max_bytes{area="heap",id="PS Eden Space"}, 2.13385216E8
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_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"}, 481527.0
jvm_buffer_memory_used_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"}, 7.1827456E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen"}, 7.08837376E8
jvm_memory_committed_bytes{area="heap",id="PS Eden Space"}, 2.10239488E8
jvm_memory_committed_bytes{area="nonheap",id="Metaspace"}, 1.63356672E8
jvm_memory_committed_bytes{area="nonheap",id="Code Cache"}, 5.4919168E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space"}, 2.4682496E7

# 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"}, 481526.0
jvm_buffer_total_capacity_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"}, 5.4466192E7
jvm_memory_used_bytes{area="heap",id="PS Old Gen"}, 9.7505448E7
jvm_memory_used_bytes{area="heap",id="PS Eden Space"}, 4.3566632E7
jvm_memory_used_bytes{area="nonheap",id="Metaspace"}, 1.47161832E8
jvm_memory_used_bytes{area="nonheap",id="Code Cache"}, 5.4194688E7
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space"}, 2.1544464E7

# 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"}, 16.0
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:
The resulting response is similar to the following:
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includedNames</td>
<td>Restricts the samples to those that match the names. Optional.</td>
</tr>
</tbody>
</table>

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_memory_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: 1110

# 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"} 7.1827456E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen"} 7.08837376E8
jvm_memory_committed_bytes{area="heap",id="PS Eden Space"} 2.10239488E8
jvm_memory_committed_bytes{area="nonheap",id="Metaspace"} 1.63356672E8
jvm_memory_committed_bytes{area="nonheap",id="Code Cache"} 5.4919168E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space"} 2.4682496E7

# 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"} 5.4466192E7
jvm_memory_used_bytes{area="heap",id="PS Old Gen"} 9.7505448E7
jvm_memory_used_bytes{area="heap",id="PS Eden Space"} 4.7526168E7
jvm_memory_used_bytes{area="nonheap",id="Metaspace"} 1.47164368E8
jvm_memory_used_bytes{area="nonheap",id="Code Cache"} 5.4247104E7
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space"} 2.154464E7
```
Chapter 20. Quartz \textit{(quartz)}

The \textit{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 \texttt{GET} request to \texttt{/actuator/quartz}, as shown in the following curl-based example:

\begin{verbatim}
$ curl 'http://localhost:8080/actuator/quartz' -i -X GET
\end{verbatim}

The resulting response is similar to the following:

\begin{verbatim}
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" ]
    }
}
\end{verbatim}

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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jobs.groups</td>
<td>Array</td>
<td>An array of job group names.</td>
</tr>
<tr>
<td>triggers.groups</td>
<td>Array</td>
<td>An array of trigger group names.</td>
</tr>
</tbody>
</table>

20.2. Retrieving Registered Job Names

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

\begin{verbatim}
$ curl 'http://localhost:8080/actuator/quartz/jobs' -i -X GET
\end{verbatim}

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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>Object</td>
<td>Job groups keyed by name.</td>
</tr>
<tr>
<td>groups.*.jobs</td>
<td>Array</td>
<td>An array of job names.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>Object</td>
<td>Trigger groups keyed by name.</td>
</tr>
<tr>
<td>groups.*.paused</td>
<td>Boolean</td>
<td>Whether this trigger group is paused.</td>
</tr>
<tr>
<td>groups.*.triggers</td>
<td>Array</td>
<td>An array of trigger names.</td>
</tr>
</tbody>
</table>

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:

```bash
$ 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:
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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>String</td>
<td>Name of the group.</td>
</tr>
<tr>
<td>jobs</td>
<td>Object</td>
<td>Job details keyed by name.</td>
</tr>
<tr>
<td>jobs.*.className</td>
<td>String</td>
<td>Fully qualified name of the job implementation.</td>
</tr>
</tbody>
</table>

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

```json
{
  "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:
<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>String</td>
<td>Name of the group.</td>
</tr>
<tr>
<td>paused</td>
<td>Boolean</td>
<td>Whether the group is paused.</td>
</tr>
<tr>
<td>triggers.cron</td>
<td>Object</td>
<td>Cron triggers keyed by name, if any.</td>
</tr>
<tr>
<td>triggers.simple</td>
<td>Object</td>
<td>Simple triggers keyed by name, if any.</td>
</tr>
<tr>
<td>triggers.dailyTimeInterval</td>
<td>Object</td>
<td>Daily time interval triggers keyed by name, if any.</td>
</tr>
<tr>
<td>triggers.calendarInterval</td>
<td>Object</td>
<td>Calendar interval triggers keyed by name, if any.</td>
</tr>
<tr>
<td>triggers.custom</td>
<td>Object</td>
<td>Any other triggers keyed by name, if any.</td>
</tr>
<tr>
<td>triggers.cron.*.previousFireTime</td>
<td>String</td>
<td>Last time the trigger fired, if any.</td>
</tr>
<tr>
<td>triggers.cron.*.nextFireTime</td>
<td>String</td>
<td>Next time at which the Trigger is scheduled to fire, if any.</td>
</tr>
<tr>
<td>triggers.cron.*.priority</td>
<td>Number</td>
<td>Priority to use if two triggers have the same scheduled fire time.</td>
</tr>
<tr>
<td>triggers.cron.*.expression</td>
<td>String</td>
<td>Cron expression to use.</td>
</tr>
<tr>
<td>triggers.cron.*.timeZone</td>
<td>String</td>
<td>Time zone for which the expression will be resolved, if any.</td>
</tr>
<tr>
<td>triggers.simple.*.previousFireTime</td>
<td>String</td>
<td>Last time the trigger fired, if any.</td>
</tr>
<tr>
<td>triggers.simple.*.nextFireTime</td>
<td>String</td>
<td>Next time at which the Trigger is scheduled to fire, if any.</td>
</tr>
<tr>
<td>triggers.simple.*.priority</td>
<td>Number</td>
<td>Priority to use if two triggers have the same scheduled fire time.</td>
</tr>
<tr>
<td>triggers.simple.*.interval</td>
<td>Number</td>
<td>Interval, in milliseconds, between two executions.</td>
</tr>
<tr>
<td>triggers.dailyTimeInterval.*.previousFireTime</td>
<td>String</td>
<td>Last time the trigger fired, if any.</td>
</tr>
<tr>
<td>triggers.dailyTimeInterval.*.nextFireTime</td>
<td>String</td>
<td>Next time at which the Trigger is scheduled to fire, if any.</td>
</tr>
<tr>
<td>triggers.dailyTimeInterval.*.priority</td>
<td>Number</td>
<td>Priority to use if two triggers have the same scheduled fire time.</td>
</tr>
<tr>
<td>triggers.dailyTimeInterval.*.interval</td>
<td>Number</td>
<td>Interval, in milliseconds, added to the fire time in order to calculate the time of the next trigger repeat.</td>
</tr>
<tr>
<td>triggers.dailyTimeInterval.*.daysOfWeek</td>
<td>Array</td>
<td>An array of days of the week upon which to fire.</td>
</tr>
<tr>
<td>triggers.dailyTimeInterval.*.startTimeOfDay</td>
<td>String</td>
<td>Time of day to start firing at the given interval, if any.</td>
</tr>
<tr>
<td>Path</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>triggers.dailyTimeInterval.*.endTimeOfDay</td>
<td>String</td>
<td>Time of day to complete firing at the given interval, if any.</td>
</tr>
<tr>
<td>triggers.calendarInterval.*.previousFireTime</td>
<td>String</td>
<td>Last time the trigger fired, if any.</td>
</tr>
<tr>
<td>triggers.calendarInterval.*.nextFireTime</td>
<td>String</td>
<td>Next time at which the Trigger is scheduled to fire, if any.</td>
</tr>
<tr>
<td>triggers.calendarInterval.*.priority</td>
<td>Number</td>
<td>Priority to use if two triggers have the same scheduled fire time.</td>
</tr>
<tr>
<td>triggers.calendarInterval.*.interval</td>
<td>Number</td>
<td>Interval, in milliseconds, added to the fire time in order to calculate the time of the next trigger repeat.</td>
</tr>
<tr>
<td>triggers.calendarInterval.*.timeZone</td>
<td>String</td>
<td>Time zone within which time calculations will be performed, if any.</td>
</tr>
<tr>
<td>triggers.custom.*.previousFireTime</td>
<td>String</td>
<td>Last time the trigger fired, if any.</td>
</tr>
<tr>
<td>triggers.custom.*.nextFireTime</td>
<td>String</td>
<td>Next time at which the Trigger is scheduled to fire, if any.</td>
</tr>
<tr>
<td>triggers.custom.*.priority</td>
<td>Number</td>
<td>Priority to use if two triggers have the same scheduled fire time.</td>
</tr>
<tr>
<td>triggers.custom.*.trigger</td>
<td>String</td>
<td>A toString representation of the custom trigger instance.</td>
</tr>
</tbody>
</table>

### 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:

```bash
$ 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:
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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>String</td>
<td>Name of the group.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the job.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the job, if any.</td>
</tr>
<tr>
<td>className</td>
<td>String</td>
<td>Fully qualified name of the job implementation.</td>
</tr>
<tr>
<td>durable</td>
<td>Boolean</td>
<td>Whether the job should remain stored after it is orphaned.</td>
</tr>
<tr>
<td>requestRecovery</td>
<td>Boolean</td>
<td>Whether the job should be re-executed if a 'recovery' or 'fail-over' situation is encountered.</td>
</tr>
<tr>
<td>Path</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>data.*</td>
<td>String</td>
<td>Job data map as key/value pairs, if any.</td>
</tr>
<tr>
<td>triggers</td>
<td>Array</td>
<td>An array of triggers associated to the job, if any.</td>
</tr>
<tr>
<td>triggers][]group</td>
<td>String</td>
<td>Name of the the trigger group.</td>
</tr>
<tr>
<td>triggers][]name</td>
<td>String</td>
<td>Name of the the trigger.</td>
</tr>
<tr>
<td>triggers][]previousFireTime</td>
<td>String</td>
<td>Last time the trigger fired, if any.</td>
</tr>
<tr>
<td>triggers][]nextFireTime</td>
<td>String</td>
<td>Next time at which the Trigger is scheduled to fire, if any.</td>
</tr>
<tr>
<td>triggers][]priority</td>
<td>Number</td>
<td>Priority to use if two triggers have the same scheduled fire time.</td>
</tr>
</tbody>
</table>

### 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:

```bash
$ 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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>String</td>
<td>Name of the group.</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the trigger.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the trigger, if any.</td>
</tr>
<tr>
<td>state</td>
<td>String</td>
<td>State of the trigger (NONE, NORMAL, PAUSED, COMPLETE, ERROR, BLOCKED).</td>
</tr>
</tbody>
</table>
### 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:
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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cron</td>
<td>Object</td>
<td>Cron trigger specific details.</td>
</tr>
<tr>
<td>cron.expression</td>
<td>String</td>
<td>Cron expression to use.</td>
</tr>
<tr>
<td>cron.timeZone</td>
<td>String</td>
<td>Time zone for which the expression will be resolved, if any.</td>
</tr>
</tbody>
</table>

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:

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

**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:

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

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

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>calendarInterval.preserveHourOfDayAcrossDaylightSavings</td>
<td>Boolean</td>
<td>Whether to fire the trigger at the same time of day, regardless of daylight saving time transitions.</td>
</tr>
<tr>
<td>calendarInterval.skipDayIfHourDoesNotExist</td>
<td>Boolean</td>
<td>Whether to skip if the hour of the day does not exist on a given day.</td>
</tr>
</tbody>
</table>
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:

```json
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@1f309d14"
   } ]
}
```
# Response Structure

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

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

```bash
$ 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": "4db5efcc-99cb-4d05-a52c-b49acfb7ea9",
      "attributeNames": [],
      "creationTime": "2021-07-22T12:48:11.816Z",
      "lastAccessedTime": "2021-07-22T17:47:34.816Z",
      "maxInactiveInterval": 1800,
      "expired": false
    },
    {
      "id": "36a188e4-ba85-40dc-8f62-caa8e7042019",
      "attributeNames": [],
      "creationTime": "2021-07-22T12:48:11.815Z",
      "maxInactiveInterval": 1800,
      "expired": false
    },
    {
      "id": "76eb7f8a-9c71-4e71-4e7e-b613-781617cea72c",
      "attributeNames": [],
      "creationTime": "2021-07-22T15:48:11.816Z",
      "lastAccessedTime": "2021-07-22T17:47:59.816Z",
      "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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Name of the user.</td>
</tr>
</tbody>
</table>

22.1.2. Response Structure

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

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

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-b49acfb7ea9",
   "attributeNames": [],
   "creationTime": "2021-07-22T12:48:11.816Z",
   "lastAccessedTime": "2021-07-22T17:47:34.816Z",
   "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:

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

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:

```sh
$ curl 'http://localhost:8080/actuator/sessions/4db5efcc-99cb-4d05-a52c-b49acfb7ea9'
-i -X DELETE
```

The preceding example deletes the session with the id of 4db5efcc-99cb-4d05-a52c-b49acfb7ea9.
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:

```bash
$ 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:

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>String</td>
<td>Message describing the result of the request.</td>
</tr>
</tbody>
</table>
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:

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

The resulting response is similar to the following:
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: 846

{
  "springBootVersion": "2.6.0-M1",
  "timeline": {
    "startTime": "2021-07-22T17:48:13.806Z",
    "events": [
      {
        "endTime": "2021-07-22T17:48:13.903Z",
        "duration": "PT0S",
        "startupStep": {
          "name": "spring.beans.instantiate",
          "id": 1,
          "tags": [
            {
              "key": "beanName",
              "value": "homeController"
            }
          ],
          "parentId": 0
        },
        "startTime": "2021-07-22T17:48:13.903Z"
      },
      {
        "endTime": "2021-07-22T17:48:13.903Z",
        "duration": "PT0.002S",
        "startupStep": {
          "name": "spring.boot.application.starting",
          "id": 0,
          "tags": [
            {
              "key": "mainApplicationClass",
              "value": "com.example.startup.StartupApplication"
            }
          ]
        },
        "startTime": "2021-07-22T17:48:13.901Z"
      }
    ]
  }
}

24.1.3. Response Structure

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

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>springBootVersion</td>
<td>String</td>
<td>Spring Boot version for this application.</td>
</tr>
<tr>
<td>timeline.startTime</td>
<td>String</td>
<td>Start time of the application.</td>
</tr>
<tr>
<td>timeline.events</td>
<td>Array</td>
<td>An array of steps collected during application startup so far.</td>
</tr>
<tr>
<td>Path</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>timeline.events[].startTime</code></td>
<td>String</td>
<td>The timestamp of the start of this event.</td>
</tr>
<tr>
<td><code>timeline.events[].endTime</code></td>
<td>String</td>
<td>The timestamp of the end of this event.</td>
</tr>
<tr>
<td><code>timeline.events[].duration</code></td>
<td>String</td>
<td>The precise duration of this event.</td>
</tr>
<tr>
<td><code>timeline.events[].startupStep.name</code></td>
<td>String</td>
<td>The name of the StartupStep.</td>
</tr>
<tr>
<td><code>timeline.events[].startupStep.id</code></td>
<td>Number</td>
<td>The id of this StartupStep.</td>
</tr>
<tr>
<td><code>timeline.events[].startupStep.parentId</code></td>
<td>Number</td>
<td>The parent id for this StartupStep.</td>
</tr>
<tr>
<td><code>timeline.events[].startupStep.tags</code></td>
<td>Array</td>
<td>An array of key/value pairs with additional step info.</td>
</tr>
<tr>
<td><code>timeline.events[].startupStep.tags[].key</code></td>
<td>String</td>
<td>The key of the StartupStep Tag.</td>
</tr>
<tr>
<td><code>timeline.events[].startupStep.tags[].value</code></td>
<td>String</td>
<td>The value of the StartupStep Tag.</td>
</tr>
</tbody>
</table>
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:

```bash
$ 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: 5948

{
  "threads" : [ {
    "threadName" : "Thread-79",
    "threadId" : 1120,
    "blockedTime" : -1,
    "blockedCount" : 0,
    "waitedTime" : -1,
    "waitedCount" : 0,
    "lockOwnerId" : -1,
    "inNative" : false,
    "suspended" : false,
    "threadState" : "RUNNABLE",
    "stackTrace" : [ ],
    "lockedMonitors" : [ ],
    "lockedSynchronizers" : [ ]
  }, {
    "threadName" : "server",
    "threadId" : 1116,
    "blockedTime" : -1,
    "blockedCount" : 0,
    "waitedTime" : -1,
    "waitedCount" : 1,
    "lockName" : "java.util.concurrent.CountDownLatch$Sync@13298fb4",
    "lockOwnerId" : -1,
    "inNative" : false,
    "suspended" : false,
    "threadState" : "WAITING",
    "stackTrace" : [ {
      "methodName" : "park",
      "fileName" : "Unsafe.java",
      "lineNumber" : 88
    }],
    "lockedMonitors" : [ ],
    "lockedSynchronizers" : [ ]
  ]
}
```
"lockedMonitors" : [ ],
"lockedSynchronizers" : [ ],
"lockInfo" : {
    "className" : "java.util.concurrent.CountDownLatch$Sync",
    "identityHashCode" : 321490868
}
}, {
    "threadName" : "pool-27-thread-1",
    "threadId" : 1106,
    "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" : "ThreadPoolExecutor.java",
        "lineNumber" : 266,
        "className" : "java.util.concurrent.ThreadPoolExecutor",
        "nativeMethod" : false
    }, {
        "methodName" : "run",
        "fileName" : "ScheduledThreadPoolExecutor.java",
        "lineNumber" : 180,
        "className" : "java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask",
        "nativeMethod" : false
    }, {
        "methodName" : "run",
        "fileName" : "ThreadPoolExecutor.java",
        "lineNumber" : 266,
        "className" : "java.util.concurrent.ThreadPoolExecutor",
        "nativeMethod" : false
    }, {
        "methodName" : "run",
        "fileName" : "ScheduledThreadPoolExecutor.java",
        "lineNumber" : 180,
        "className" : "java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask",
        "nativeMethod" : false
    } ]
}
"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" : 748,
"className" : "java.lang.Thread",
"nativeMethod" : false
}
],
"lockedMonitors" : [ {
"className" : "java.lang.Object",
"identityHashCode" : 799872910,
"lockedStackDepth" : 0,
"lockedStackFrame" : {
"methodName" : "schedule",
"fileName" : "ReschedulingRunnable.java",
"lineNumber" : 76,
"className" :
"org.springframework.scheduling.concurrent.ReschedulingRunnable",
"nativeMethod" : false
}
},
"className" : "java.lang.Object",
"identityHashCode" : 799872910,
"lockedStackDepth" : 1,
"lockedStackFrame" : {
"methodName" : "run",
"fileName" : "ReschedulingRunnable.java",
"lineNumber" : 101,
"className" :
"org.springframework.scheduling.concurrent.ReschedulingRunnable",
"nativeMethod" : false
}
}
],
"lockedSynchronizers" : [ {
"className" : "java.util.concurrent.ThreadPoolExecutor$Worker",
"nativeMethod" : false
} 
]
25.1.1. Response Structure

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

<table>
<thead>
<tr>
<th>Path</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>threads</td>
<td>Array</td>
<td>JVM’s threads.</td>
</tr>
<tr>
<td>threads.[].blockedCount</td>
<td>Number</td>
<td>Total number of times that the thread has been blocked.</td>
</tr>
<tr>
<td>threads.[].blockedTime</td>
<td>Number</td>
<td>Time in milliseconds that the thread has spent blocked. -1 if thread contention monitoring is disabled.</td>
</tr>
<tr>
<td>threads.[].daemon</td>
<td>Boolean</td>
<td>Whether the thread is a daemon thread. Only available on Java 9 or later.</td>
</tr>
<tr>
<td>threads.[].inNative</td>
<td>Boolean</td>
<td>Whether the thread is executing native code.</td>
</tr>
<tr>
<td>threads.[].lockName</td>
<td>String</td>
<td>Description of the object on which the thread is blocked, if any.</td>
</tr>
<tr>
<td>threads.[].lockInfo</td>
<td>Object</td>
<td>Object for which the thread is blocked waiting.</td>
</tr>
<tr>
<td>threads.[].lockInfo.className</td>
<td>String</td>
<td>Fully qualified class name of the lock object.</td>
</tr>
<tr>
<td>threads.[].lockInfo.identityHashCode</td>
<td>Number</td>
<td>Identity hash code of the lock object.</td>
</tr>
<tr>
<td>threads.[].lockedMonitors</td>
<td>Array</td>
<td>Monitors locked by this thread, if any</td>
</tr>
<tr>
<td>threads.[].lockedMonitors.[].className</td>
<td>String</td>
<td>Class name of the lock object.</td>
</tr>
<tr>
<td>threads.[].lockedMonitors.[].identityHashCode</td>
<td>Number</td>
<td>Identity hash code of the lock object.</td>
</tr>
<tr>
<td>threads.[].lockedMonitors.[].lockedStackDepth</td>
<td>Number</td>
<td>Stack depth where the monitor was locked.</td>
</tr>
<tr>
<td>threads.[].lockedMonitors.[].lockedStackFrame</td>
<td>Object</td>
<td>Stack frame that locked the monitor.</td>
</tr>
<tr>
<td>Path</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>threads.[].lockedSynchronizers</td>
<td>Array</td>
<td>Synchronizers locked by this thread.</td>
</tr>
<tr>
<td>threads.[].lockedSynchronizers[].className</td>
<td>String</td>
<td>Class name of the locked synchronizer.</td>
</tr>
<tr>
<td>threads.[].lockedSynchronizers[].identityHashCode</td>
<td>Number</td>
<td>Identity hash code of the locked synchronizer.</td>
</tr>
<tr>
<td>threads.[].lockOwnerId</td>
<td>Number</td>
<td>ID of the thread that owns the object on which the thread is blocked. -1 if the thread is not blocked.</td>
</tr>
<tr>
<td>threads.[].lockOwnerName</td>
<td>String</td>
<td>Name of the thread that owns the object on which the thread is blocked, if any.</td>
</tr>
<tr>
<td>threads.[].priority</td>
<td>Number</td>
<td>Priority of the thread. Only available on Java 9 or later.</td>
</tr>
<tr>
<td>threads.[].stackTrace</td>
<td>Array</td>
<td>Stack trace of the thread.</td>
</tr>
<tr>
<td>threads.[].stackTrace[].classLoaderName</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>threads.[].stackTrace[].className</td>
<td>String</td>
<td>Name of the class that contains the execution point identified by this entry.</td>
</tr>
<tr>
<td>threads.[].stackTrace[].fileName</td>
<td>String</td>
<td>Name of the source file that contains the execution point identified by this entry, if any.</td>
</tr>
<tr>
<td>threads.[].stackTrace[].lineNumber</td>
<td>Number</td>
<td>Line number of the execution point identified by this entry. Negative if unknown.</td>
</tr>
<tr>
<td>threads.[].stackTrace[].methodName</td>
<td>String</td>
<td>Name of the method.</td>
</tr>
<tr>
<td>threads.[].stackTrace[].moduleName</td>
<td>String</td>
<td>Name of the module that contains the execution point identified by this entry, if any. Only available on Java 9 or later.</td>
</tr>
<tr>
<td>threads.[].stackTrace[].moduleVersion</td>
<td>String</td>
<td>Version of the module that contains the execution point identified by this entry, if any. Only available on Java 9 or later.</td>
</tr>
<tr>
<td>Path</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>threads[].stackTrace[].nativeMethod</td>
<td>Boolean</td>
<td>Whether the execution point is a native method.</td>
</tr>
<tr>
<td>threads[].suspended</td>
<td>Boolean</td>
<td>Whether the thread is suspended.</td>
</tr>
<tr>
<td>threads[].threadId</td>
<td>Number</td>
<td>ID of the thread.</td>
</tr>
<tr>
<td>threads[].threadName</td>
<td>String</td>
<td>Name of the thread.</td>
</tr>
<tr>
<td>threads[].threadState</td>
<td>String</td>
<td>State of the thread (NEW, RUNNABLE, BLOCKED, WAITING, TIMED_WAITING, TERMINATED).</td>
</tr>
<tr>
<td>threads[].waitedCount</td>
<td>Number</td>
<td>Total number of times that the thread has waited for notification.</td>
</tr>
<tr>
<td>threads[].waitedTime</td>
<td>Number</td>
<td>Time in milliseconds that the thread has spent waiting. -1 if thread contention monitoring is disabled</td>
</tr>
</tbody>
</table>

### 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: 66480

2021-07-22 17:48:32
Full thread dump OpenJDK 64-Bit Server VM (25.292-b10 mixed mode):

"server" - Thread t@1116
   java.lang.Thread.State: WAITING
   at sun.misc.Unsafe.park(Native Method)
   - parking to wait for <13298fb4> (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
```
Locked ownable synchronizers:
  - None

"pool-27-thread-1" - Thread t@1106
  java.lang.Thread.State: RUNNABLE
  at java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.siftUp(ScheduledThreadPoolExecutor.java:886)
  at java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.offer(ScheduledThreadPoolExecutor.java:1020)
  at java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.add(ScheduledThreadPoolExecutor.java:1037)
  at java.util.concurrent.ScheduledThreadPoolExecutor.delayedExecute(ScheduledThreadPoolExecutor.java:328)
  at java.util.concurrent.ScheduledThreadPoolExecutor.schedule(ScheduledThreadPoolExecutor.java:533)
  at java.util.concurrent.Executors$DelegatedScheduledExecutorService.schedule(Executors.java:729)
  at org.springframework.scheduling.concurrent.ReschedulingRunnable.schedule(ReschedulingRunnable.java:82)
  - locked <2fad178e> (a java.lang.Object)
  at org.springframework.scheduling.concurrent.ReschedulingRunnable.run(ReschedulingRunnable.java:101)
  - locked <2fad178e> (a java.lang.Object)
  at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
  at java.util.concurrent.FutureTask.run(FutureTask.java:266)
java.lang.Thread.State: RUNNABLE
    at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
    at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
        - locked <42e136a0> (a sun.nio.ch.Util$3)
        - locked <5e9ea57f> (a java.util.Collections$UnmodifiableSet)
        - locked <6598bb37> (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:788)
    at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
    - None

"http-nio-auto-41-exec-10" - Thread t@1087
    java.lang.Thread.State: WAITING
        at sun.misc.Unsafe.park(Native Method)
        - parking to wait for <5a17af53> (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$Worker$Runnable.run(TaskQueue.java:61)
    at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
    - None

"http-nio-auto-41-exec-9" - Thread t@1086
    java.lang.Thread.State: WAITING
        at sun.misc.Unsafe.park(Native Method)
        - parking to wait for <5a17af53> (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$Worker$Runnable.run(TaskQueue.java:61)
    at java.lang.Thread.run(Thread.java:748)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
- None

"http-nio-auto-41-exec-8" - Thread t@1085
java.lang.Thread.State: WAITING
at sun.misc.Unsafe.park(Native Method)
- parking to wait for <5a17af53> (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 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:61)
at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
- None

"http-nio-auto-41-exec-7" - Thread t@1084
java.lang.Thread.State: WAITING
at sun.misc.Unsafe.park(Native Method)
- parking to wait for <5a17af53> (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 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:61)
at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
- None
"http-nio-auto-41-exec-6" - Thread t@1083
java.lang.Thread.State: WAITING
  at sun.misc.Unsafe.park(Native Method)
- parking to wait for <5a17af53> (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 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:61)
at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
- None

"http-nio-auto-41-exec-5" - Thread t@1082
java.lang.Thread.State: WAITING
  at sun.misc.Unsafe.park(Native Method)
- parking to wait for <5a17af53> (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 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:61)
at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
- None

"http-nio-auto-41-exec-4" - Thread t@1081
java.lang.Thread.State: WAITING
  at sun.misc.Unsafe.park(Native Method)
- parking to wait for <5a17af53> (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 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:61)
at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
- None

"http-nio-auto-41-exec-3" - Thread t@1080
java.lang.Thread.State: WAITING
at sun.misc.Unsafe.park(Native Method)
- parking to wait for <5a17af53> (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 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:61)
at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
- None

"http-nio-auto-41-exec-2" - Thread t@1079
java.lang.Thread.State: WAITING
at sun.misc.Unsafe.park(Native Method)
- parking to wait for <5a17af53> (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 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:61)
at java.lang.Thread.run(Thread.java:748)
Locked ownable synchronizers:
- None

"http-nio-auto-41-exec-1" - Thread t@1078
java.lang.Thread.State: WAITING
at sun.misc.Unsafe.park(Native Method)
- parking to wait for <5a17af53> (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 java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1074)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
- None

"Catalina-utility-2" - Thread t@1077
java.lang.Thread.State: WAITING
at sun.misc.Unsafe.park(Native Method)
- parking to wait for <1188c687> (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:61)
Locked ownable synchronizers:
  - None

"container-0" - Thread t@1076
  java.lang.Thread.State: TIMED_WAITING
  at java.lang.Thread.sleep(Native Method)
  at org.apache.catalina.core.StandardServer.await(StandardServer.java:570)
  at org.springframework.boot.web.embedded.tomcat.TomcatWebServer$1.run(TomcatWebServer.java:197)

Locked ownable synchronizers:
  - None

"Catalina-utility-1" - Thread t@1075
  java.lang.Thread.State: TIMED_WAITING
  at sun.misc.Unsafe.park(Native Method)
  - parking to wait for <1188c687> (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:61)
    at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
  - None

"reactor-http-epoll-8" - Thread t@794
  java.lang.Thread.State: RUNNABLE
  at io.netty.channel.epoll.Native.epollWait(Native Method)
  at io.netty.channel.epoll.Native.epollWait(Native.java:192)
  at io.netty.channel.epoll.Native.epollWait(Native.java:185)
  at io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:290)
  at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:347)
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:748)

Locked ownable synchronizers:
  - None

"reactor-http-epoll-7" - Thread t@793
    java.lang.Thread.State: RUNNABLE
    at io.netty.channel.epoll.Native.epollWait(Native Method)
    at io.netty.channel.epoll.Native.epollWait(Native.java:192)
    at io.netty.channel.epoll.Native.epollWait(Native.java:185)
    at io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:290)
    at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:347)
    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:748)

Locked ownable synchronizers:
  - None

"reactor-http-epoll-6" - Thread t@792
    java.lang.Thread.State: RUNNABLE
    at io.netty.channel.epoll.Native.epollWait(Native Method)
    at io.netty.channel.epoll.Native.epollWait(Native.java:192)
    at io.netty.channel.epoll.Native.epollWait(Native.java:185)
    at io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:290)
    at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:347)
    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:748)

Locked ownable synchronizers:
  - None

"reactor-http-epoll-5" - Thread t@791
    java.lang.Thread.State: RUNNABLE
    at io.netty.channel.epoll.Native.epollWait(Native Method)
    at io.netty.channel.epoll.Native.epollWait(Native.java:192)
Locked ownable synchronizers:
- None

"reactor-http-epoll-4" - Thread t@790
java.lang.Thread.State: RUNNABLE
  at io.netty.channel.epoll.Native.epollWait(Native Method)
  at io.netty.channel.epoll.Native.epollWait(Native.java:192)
  at io.netty.channel.epoll.Native.epollWait(Native.java:185)
  at io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:290)
  at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:347)
  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:748)

Locked ownable synchronizers:
- None

"reactor-http-epoll-3" - Thread t@789
java.lang.Thread.State: RUNNABLE
  at io.netty.channel.epoll.Native.epollWait(Native Method)
  at io.netty.channel.epoll.Native.epollWait(Native.java:192)
  at io.netty.channel.epoll.Native.epollWait(Native.java:185)
  at io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:290)
  at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:347)
  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:748)

Locked ownable synchronizers:
- None
"reactor-http-epoll-2" - Thread t@788
java.lang.Thread.State: RUNNABLE
  at io.netty.channel.epoll.Native.epollWait(Native Method)
  at io.netty.channel.epoll.Native.epollWait(Native.java:192)
  at io.netty.channel.epoll.Native.epollWait(Native.java:185)
  at io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:290)
  at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:347)
  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:748)

Locked ownable synchronizers:  
  - None

"reactor-http-epoll-1" - Thread t@787
java.lang.Thread.State: RUNNABLE
  at io.netty.channel.epoll.Native.epollWait(Native Method)
  at io.netty.channel.epoll.Native.epollWait(Native.java:192)
  at io.netty.channel.epoll.Native.epollWait(Native.java:185)
  at io.netty.channel.epoll.EpollEventLoop.epollWaitNoTimerChange(EpollEventLoop.java:290)
  at io.netty.channel.epoll.EpollEventLoop.run(EpollEventLoop.java:347)
  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:748)

Locked ownable synchronizers:  
  - None

"boundedElastic-2" - Thread t@785
java.lang.Thread.State: WAITING
  at sun.misc.Unsafe.park(Native Method)
    - parking to wait for <3a963c94> (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$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:748)
PoolExecutor.java:1081)
   at java.util.concurrent.ScheduledThreadPoolExecutor$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:748)

Locked ownable synchronizers:
- None

"boundedElastic-1" - Thread t@736
   java.lang.Thread.State: WAITING
   at sun.misc.Unsafe.park(Native Method)
   - parking to wait for <1af779f2> (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$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1081)
   at java.util.concurrent.ScheduledThreadPoolExecutor$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:748)

Locked ownable synchronizers:
- None

"Keep-Alive-Timer" - Thread t@675
   java.lang.Thread.State: TIMED_WAITING
   at java.lang.Thread.sleep(Native Method)
   at java.lang.Thread.run(Thread.java:748)

Locked ownable synchronizers:
- None

"parallel-8" - Thread t@415
   java.lang.Thread.State: WAITING
   at sun.misc.Unsafe.park(Native Method)
   - parking to wait for <620f9e5f> (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$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1081)
at java.util.concurrent.ScheduledThreadPoolExecutor$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:748)

Locked ownable synchronizers:
- None

"boundedElastic-evictor-1" - Thread t@377
  java.lang.Thread.State: TIMED_WAITING
  at sun.misc.Unsafe.park(Native Method)
  - parking to wait for <25dfea6f> (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:748)

Locked ownable synchronizers:
- None

"parallel-7" - Thread t@376
  java.lang.Thread.State: WAITING
  at sun.misc.Unsafe.park(Native Method)
  - parking to wait for <43b69650> (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:1081)
Locked ownable synchronizers:
- None

"nioEventLoopGroup-10-2" - Thread t@282
java.lang.Thread.State: RUNNABLE
at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
- locked <4bd3ad3> (a io.netty.channel.nio.SelectedSelectionKeySet)
- locked <3c7e31e4> (a java.util.Collections$UnmodifiableSet)
- locked <76ea9cc4> (a sun.nio.ch.EPollSelectorImpl)
at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:97)
at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:101)
at io.netty.channel.nio.SelectedSelectionKeySetSelector.select(SelectedSelectionKeySetSelector.java:68)
at io.netty.channel.nio.NioEventLoop.select(NioEventLoop.java:810)
at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:457)
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:748)

Locked ownable synchronizers:
- None

"nioEventLoopGroup-10-1" - Thread t@275
java.lang.Thread.State: RUNNABLE
at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
- locked <3ec9f782> (a io.netty.channel.nio.SelectedSelectionKeySet)
- locked <1c30192a> (a java.util.Collections$UnmodifiableSet)
- locked <510cfa59> (a sun.nio.ch.EPollSelectorImpl)
at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:97)
at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:101)
at io.netty.channel.nio.SelectedSelectionKeySetSelector.select(SelectedSelectionKeySetSelector.java:68)
at io.netty.channel.nio.NioEventLoop.select(NioEventLoop.java:810)
at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:457)
Locked ownable synchronizers:
- None

"nioEventLoopGroup-6-1" - Thread t@228
java.lang.Thread.State: RUNNABLE
  at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
  at sun.nio.ch.SelectorImpl.doSelect(SelectorImpl.java:93)
  at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
    - locked <7bbd8e71> (a io.netty.channel.nio.SelectedSelectionKeySet)
    - locked <4e0e56> (a java.util.Collections$UnmodifiableSet)
    - locked <6bba663> (a sun.nio.ch.EPollSelectorImpl)
  at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:97)
  at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:101)
  at io.netty.channel.nio.SelectedSelectionKeySetSelector.select(SelectedSelectionKeySetSelector.java:68)
    at io.netty.channel.nio.NioEventLoop.select(NioEventLoop.java:810)
    at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:457)
  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:748)

Locked ownable synchronizers:
- None

"nioEventLoopGroup-6-2" - Thread t@227
java.lang.Thread.State: RUNNABLE
  at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
    - locked <2c73fa33> (a io.netty.channel.nio.SelectedSelectionKeySet)
    - locked <382927a> (a java.util.Collections$UnmodifiableSet)
    - locked <4441795> (a sun.nio.ch.EPollSelectorImpl)
  at
Locked ownable synchronizers:
- None

"nioEventLoopGroup-4-2" - Thread t@160
java.lang.Thread.State: RUNNABLE
at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
- locked <179b560f> (a io.netty.channel.nio.SelectedSelectionKeySet)
- locked <240a84ad> (a java.util.Collections$UnmodifiableSet)
- locked <556cf091> (a sun.nio.ch.EPollSelectorImpl)
at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:101)
at io.netty.channel.nio.SelectedSelectionKeySetSelector.select(SelectedSelectionKeySetSelector.java:68)
at io.netty.channel.nio.NioEventLoop.select(NioEventLoop.java:810)
at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:457)
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:748)

Locked ownable synchronizers:
- None

"nioEventLoopGroup-4-1" - Thread t@159
java.lang.Thread.State: RUNNABLE
at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
- locked <1a28c8cf> (a io.netty.channel.nio.SelectedSelectionKeySet)
- locked <574eceed> (a java.util.Collections$UnmodifiableSet)
- locked <27910703> (a sun.nio.ch.EPollSelectorImpl)
Locked ownable synchronizers:
- None

"parallel-4" - Thread t@113
java.lang.Thread.State: WAITING
at sun.misc.Unsafe.park(Native Method)
- parking to wait for <15cb8318> (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$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1081)
at java.util.concurrent.ScheduledThreadPoolExecutor$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:748)

Locked ownable synchronizers:
- None

"nioEventLoopGroup-2-1" - Thread t@68
java.lang.Thread.State: RUNNABLE
at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
- locked <4f9f5fde> (a io.netty.channel.nio.SelectedSelectionKeySet)
- locked <68e3fbd6> (a java.util.Collections$UnmodifiableSet)
- locked <476540dc> (a sun.nio.ch.EPollSelectorImpl)
at sun.nio.ch.SelectorImpl.select(SelectorImpl.java:97)
Locked ownable synchronizers:
- None

"parallel-2" - Thread t@15
  java.lang.Thread.State: WAITING
  at sun.misc.Unsafe.park(Native Method)
  - parking to wait for <67f3349b> (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$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1081)
  at java.util.concurrent.ScheduledThreadPoolExecutor$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:748)

Locked ownable synchronizers:
- None

"parallel-1" - Thread t@14
  java.lang.Thread.State: WAITING
  at sun.misc.Unsafe.park(Native Method)
  - parking to wait for <315094f9> (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$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1081)
  at java.util.concurrent.ScheduledThreadPoolExecutor$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$Worker.run(ThreadPoolExecutor.java:624)
  at java.lang.Thread.run(Thread.java:748)
at java.util.concurrent.ScheduledThreadPoolExecutor$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:748)

Locked ownable synchronizers:
  - None

"/127.0.0.1:52500 to /127.0.0.1:37559 workers Thread 3" - Thread t@13
java.lang.Thread.State: RUNNABLE
  at sun.nio.ch.EPollArrayWrapper.epollWait(Native Method)
  at sun.nio.ch.SelectorImpl.lockAndDoSelect(SelectorImpl.java:86)
- locked <4124b850> (a sun.nio.ch.Util$3)
- locked <2aa479c1> (a java.util.Collections$UnmodifiableSet)
- locked <540dc7e2> (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$SocketInputStream.read(SocketConnection.java:185)
  at com.esotericsoftware.kryo.io.Input.fill(Input.java:146)
  at com.esotericsoftware.kryo.io.Input.require(Input.java:178)
  at com.esotericsoftware.kryo.io.Input.readByte(Input.java:295)
  at org.gradle.internal.serialize.kryo.KryoBackedDecoder.readByte(KryoBackedDecoder.java:82)
  at org.gradle.internal.remote.internal.hub.InterHubMessageSerializer$MessageReader.read(InterHubMessageSerializer.java:64)
  at org.gradle.internal.remote.internal.hub.InterHubMessageSerializer$MessageReader.read(InterHubMessageSerializer.java:52)
  at org.gradle.internal.remote.internal.inet.SocketConnection.receive(SocketConnection.java:81)
  at org.gradle.internal.remote.internal.hub.MessageHub$ConnectionReceive.run(MessageHub.java:270)
  at org.gradle.internal.concurrent.ExecutorPolicy$CatchAndRecordFailures.onExecute(ExecutorPolicy.java:64)
  at org.gradle.internal.concurrent.ManagedExecutorImpl$1.run(ManagedExecutorImpl.java:48)
  at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
  at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
Locked ownable synchronizers:
- Locked <7ee955a8> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"/127.0.0.1:52500 to /127.0.0.1:37559 workers Thread 2" - Thread t@12
java.lang.Thread.State: WAITING
  - parking to wait for <614f76c6> (a java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject)
  - Locked <7ee955a8> (a java.util.concurrent.ThreadPoolExecutor$Worker)

"Test worker" - Thread t@11
java.lang.Thread.State: RUNNABLE
  - Locked <51e5fc98> (a java.util.concurrent.ThreadPoolExecutor$Worker)
at org.springframework.test.web.servlet.TestDispatcherServlet.service(TestDispatcherServlet.java:72)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:750)
at org.springframework.mock.web.MockFilterChain$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:183)
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:688)
at org.junit.jupiter.engine.execution.MethodInvocation.proceed(MethodInvocation.java:60)
at org.junit.jupiter.engine.extension.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:149)
at org.junit.jupiter.engine.extension.TimeoutExtension.interceptTestMethod(TimeoutExtension.java:84)
at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor$$Lambda$129/1732170861.apply(Unknown Source)
at org.junit.jupiter.engine.execution.ExecutableInvoker$ReflectiveInterceptorCall.lambda$ofVoidMethod$0(ExecutableInvoker.java:115)
at org.junit.jupiter.engine.execution.ExecutableInvoker$ReflectiveInterceptorCall$$Lambda$130/1223355872.apply(Unknown Source)
at org.junit.jupiter.engine.execution.InvocationInterceptorChain$InterceptedInvocation.proceed(InvocationInterceptorChain.java:106)
at org.junit.jupiter.engine.execution.InvocationInterceptorChain.proceed(InvocationInterceptorChain.java:64)
at org.junit.jupiter.engine.execution.InvocationInterceptorChain.chainAndInvoke(InvocationInterceptorChain.java:45)
at org.junit.jupiter.engine.execution.InvocationInterceptorChain.invoke(InvocationInterceptorChain.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$6(TestMethodTestDescriptor.java:210)
at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor$$Lambda$289/155290271.execute(Unknown Source)
at org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.execute(TestMethodTestDescriptor.java:131)
at org.junit.jupiter.engine.descriptor.TestMethodTestDescriptor.execute(TestMethodTestDescriptor.java:65)
at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$5(NodeTestTask.java:139)
at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$199/667739354.execute(Unknown Source)
at org.junit.platform.engine.support.hierarchical.ThrowableCollector.execute(ThrowableCollector.java:73)
at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$7(NodeTestTask.java:129)
at org.junit.platform.engine.support.hierarchical.NodeTestTask$$Lambda$198/1187497517.invoke(Unknown Source)
at org.junit.platform.engine.support.hierarchical.Node.around(Node.java:137)
at org.junit.platform.engine.support.hierarchical.NodeTestTask.lambda$executeRecursively$8(NodeTestTask.java:127)
at java.lang.ref.Reference.tryHandlePending(Reference.java:191)
    at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:153)

Locked ownable synchronizers:
  - None