

# 一. 概述

## 1.1 介绍

Sentinel 引入了 Sentinel API Gateway Adapter Common 模块，此模块中包含**网关限流的规则**和**自定义 API** 的实体和管理逻辑：

- GatewayFlowRule：网关限流规则，针对 API Gateway 的场景定制的限流规则，可以针对不同 **route\_id**或**自定义的 API** 分组进行限流，支持针对请求中的**参数**、**Header**、**来源 IP** 等进行定制化的限流。
- ApiDefinition：**用户自定义的 API 定义分组**，可以看做是一些 URL 匹配的组合。比如我们可以定义一个 API 叫 my\_api，请求 path 模式为 /foo/\*\* 和 /baz/\*\* 的都归到 my\_api 这个 API 分组下面。限流的时候可以针对这个自定义的 API 分组维度进行限流。

## 1.2 参数说明

其中网关限流规则 GatewayFlowRule 的字段解释如下：

- resource：资源名称，可以是网关中的 route 名称或者用户自定义的 API 分组名称。
- resourceMode：规则是针对 API Gateway 的 route (RESOURCE\_MODE\_ROUTE\_ID) 还是用户在 Sentinel 中定义的 API 分组 (RESOURCE\_MODE\_CUSTOM\_API\_NAME)，默认是 route。
- grade：限流指标维度，同限流规则的 grade 字段
- count：限流阈值
- intervalSec：统计时间窗口，单位是秒，默认是 1 秒
- controlBehavior：**流量整形的控制效果**，同限流规则的 controlBehavior 字段，目前支持快速失败和匀速排队两种模式，默认是快速失败。
- burst：应对突发请求时额外允许的请求数目。
- maxQueueingTimeoutMs：匀速排队模式下的最长排队时间，单位是毫秒，仅在匀速排队模式下生效。
- paramItem：参数限流配置。若不提供，则代表不针对参数进行限流，该网关规则将会被转换成普通流控规则；否则会转换成热点规则。其中的字段：
- parseStrategy：从请求中提取参数的策略，目前支持提取来源 IP (PARAM\_PARSE\_STRATEGY\_CLIENT\_IP)、Host (PARAM\_PARSE\_STRATEGY\_HOST)、任意 Header (PARAM\_PARSE\_STRATEGY\_HEADER) 和任意 URL 参数 (PARAM\_PARSE\_STRATEGY\_URL\_PARAM) 四种模式。
- fieldName：若提取策略选择 Header 模式或 URL 参数模式，则需要指定对应的 header 名称或 URL 参数名称。
- pattern 和 matchStrategy：为后续参数匹配特性预留，目前未实现。

## 相关参数

```
public final class SentinelGatewayConstants {
    public static final int APP_TYPE_GATEWAY = 1;
    public static final int RESOURCE_MODE_ROUTE_ID = 0;
    public static final int RESOURCE_MODE_CUSTOM_API_NAME = 1;
    public static final int PARAM_PARSE_STRATEGY_CLIENT_IP = 0;
    public static final int PARAM_PARSE_STRATEGY_HOST = 1;
    public static final int PARAM_PARSE_STRATEGY_HEADER = 2;
```

```
public static final int PARAM_PARSE_STRATEGY_URL_PARAM = 3;
public static final int PARAM_PARSE_STRATEGY_COOKIE = 4;
public static final int URL_MATCH_STRATEGY_EXACT = 0;
public static final int URL_MATCH_STRATEGY_PREFIX = 1;
public static final int URL_MATCH_STRATEGY_REGEX = 2;
public static final int PARAM_MATCH_STRATEGY_EXACT = 0;
public static final int PARAM_MATCH_STRATEGY_PREFIX = 1;
public static final int PARAM_MATCH_STRATEGY_REGEX = 2;
public static final int PARAM_MATCH_STRATEGY_CONTAINS = 3;
}
```

### 1.3 网关限流参数组合说明:

#### 限流参数组合1: (按照QPS限流:快速失败)

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、阈值类型: QPS
- 4、QPS阈值: 阈值
- 5、间隔时间: 秒、时、分
- 6、流控方式: 快速失败
- 7、Burst size: 0

#### 限流参数组合2: (按照QPS限流:匀速排队)

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、阈值类型: QPS
- 4、QPS阈值: 阈值
- 5、间隔时间: 秒、时、分
- 6、流控方式: 匀速排队
- 7、超时时间: 毫秒

#### 限流参数组合3: (按照线程数限流)

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、阈值类型: 线程数
- 4、线程数: 阈值

#### 限流参数组合4: (按照请求参数限流: Client IP\Remote Host)

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、请求参数: Client IP\Remote Host

IP: 0 HOST:1

```
{
  "resource": "httpbin_route",
  "count": 0,
  "paramItem": {
    "parseStrategy": 0
  }
}
```

#### 限流参数组合5: (按照请求参数限流: Header)

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、请求参数: Header
- 4、Header名称: header名称

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、请求参数: Header
- 4、Header名称: header名称
- 5、属性值匹配: 精确、子串、正则
- 6、匹配串: 匹配串

```
{
  "resource": "httpbin_route",
  "count": 0,
  "paramItem": {
    "parseStrategy": 2,
    "fieldName": "Spring-Cloud-Alibaba"
  }
}
```

#### 限流参数组合6: (按照请求参数限流: URL参数)

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、请求参数: Header
- 4、URL参数: URL参数

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、请求参数: Header
- 4、URL参数: URL参数
- 5、属性值匹配: 精确、子串、正则
- 6、匹配串: 匹配串

```
{
  "resource": "httpbin_route",
  "count": 0,
  "paramItem": {
    "parseStrategy": 3,
    "fieldName": "name"
  }
}
```

#### 限流参数组合7: (按照请求参数限流: Cookie)

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、请求参数: Header
- 4、Cookie: Cookie

- 1、API类型: RoutID\API分组
- 2、API名称: RoutID\API分组名称
- 3、请求参数: Header
- 4、Cookie: Cookie
- 5、属性值匹配: 精确、子串、正则
- 6、匹配串: 匹配串

### 1.4 网关降级参数组合说明:

#### 降级参数组合1: (按照相应时间)

- 1、资源名: 资源名
- 2、降级策略: RT
- 3、RT: 毫秒
- 4、时间窗口: 降级时间间隔, 单位秒

#### 降级参数组合2: (按照异常比率)

- 1、资源名: 资源名
- 2、降级策略: RT
- 3、按照异常比率: 0.0~1.0
- 4、时间窗口: 降级时间间隔, 单位秒

#### 降级参数组合3: (按照异常数)

- 1、资源名: 资源名
- 2、降级策略: RT
- 3、异常数: 异常数
- 4、时间窗口: 降级时间间隔, 单位秒

## 1.5、限流类型 RuleType

```
public enum RuleType {

    /**
     * flow.
     */
    FLOW("flow", FlowRule.class),
    /**
     * degrade.
     */
    DEGRADE("degrade", DegradeRule.class),
    /**
     * param flow.
     */
    PARAM_FLOW("param-flow", ParamFlowRule.class),
    /**
     * system.
     */
    SYSTEM("system", SystemRule.class),
    /**
     * authority.
     */
    AUTHORITY("authority", AuthorityRule.class),
    /**
     * gateway flow.
     */
    GW_FLOW("gw-flow",
            "com.alibaba.csp.sentinel.adapter.gateway.common.rule.GatewayFlowRule"),
    /**
     * api.
     */
    GW_API_GROUP("gw-api-group",
            "com.alibaba.csp.sentinel.adapter.gateway.common.api.ApiDefinition");
}
```

## 1.6 当前版本如下:

```
nacos: 1.1.4 或 1.2.1
sentinel: 1.6.3
springboot: 2.0.6.RELEASE
springCloud: Finchley.SR2
```

# 二、项目配置

## 2.1 网关配置

pom.xml

```
<!-- nacos 注册中心 -->
<dependency>
    <groupId>com.timeloit.cloud</groupId>
    <artifactId>spring-cloud-starter-timeloit-nacos-discovery</artifactId>
</dependency>

<!-- sentinel gateway流控 -->
<dependency>
    <groupId>com.timeloit.cloud</groupId>
    <artifactId>spring-cloud-starter-timeloit-sentinel</artifactId>
</dependency>

<dependency>
    <groupId>com.timeloit.cloud</groupId>
    <artifactId>spring-cloud-timeloit-sentinel-gateway</artifactId>
</dependency>

<!-- sentinel nacos 整合 -->
<dependency>
    <groupId>com.timeloit.cloud</groupId>
    <artifactId>spring-cloud-timeloit-sentinel-datasource</artifactId>
</dependency>

<dependency>
    <groupId>com.alibaba.csp</groupId>
    <artifactId>sentinel-datasource-nacos</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
```

## 2.2 配置文件

application.yml

```
spring:
  application:
    name: gateway-service
```

```

main:
  allow-bean-definition-overriding: true
cloud:
  # 使用 Nacos 作为服务注册发现
  nacos.discovery:
    server-addr: 47.114.50.99:8010
    namespace: 46c2400a-5773-4c26-be68-2e90a673259b
  gateway:
    enabled: true
    discovery:
      locator:
        lower-case-service-id: true
    routes:
      - id: loit-portal-id
        uri: lb://loit-portal
        predicates:
          - Path=/api-portal/**
        filters:
          - StripPrefix=1
        logoutSign: true
    sentinel:
# 配置文件存储在本地
#   datasource.ds2.file:
#     file: "classpath: gateway-service-sentinel-gateway"
#     ruleType: gw-flow
#   datasource.ds1.file:
#     file: "classpath: gateway-service-sentinel-api"
#     ruleType: gw-api-group
  enabled: true
  datasource:
    ds2:
      nacos:
        data-id: ${spring.application.name}-sentinel-gateway
        group-id: DEFAULT_GROUP
        rule-type: gw-flow
        server-addr: 47.114.50.99:8010
        namespace: 46c2400a-5773-4c26-be68-2e90a673259b
        data-type: json

    ds1:
      nacos:
        data-id: ${spring.application.name}-sentinel-api
        group-id: DEFAULT_GROUP
        rule-type: gw-api-group
        server-addr: 47.114.50.99:8010
        namespace: 46c2400a-5773-4c26-be68-2e90a673259b
        data-type: json

## 应用与Sentinel控制台交互的端口，应用本地会起一个该端口占用的HttpServer
transport:
  ## Sentinel 控制台地址
  dashboard: localhost:8080

  ## 应用与Sentinel控制台的心跳间隔时间

```

```
heartbeat-interval-ms: 60000
filter:
  enabled: true
scg.fallback:
  ## Spring Cloud Gateway 熔断后的响应模式 (选择 redirect or response)
  mode: response
  ## Spring Cloud Gateway 响应模式为 'response' 模式对应的响应码
  response-status: 444
  ## Spring Cloud Gateway 响应模式为 'response' 模式对应的响应内容
  response-body: 系统繁忙请稍后再试
scg:
  order: -100
```

### 三、限流规则使用说明

public | loit-test | loit-other | [chenshiying](#)

配置管理 | [chenshiying](#) 46c2400a-5773-4c26-be68-2e90a673259b 查询结果: 共查询到 3 条满足要求的配置。

Data ID:  Group:  [查询](#) [高级查询](#) [导出查询结果](#) [导入配置](#)

<input type="checkbox"/>	Data Id	Group	归属应用:	操作
<input type="checkbox"/>	gateway-service-sentinel-gateway	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	gateway-service-sentinel-api	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>
<input type="checkbox"/>	gateway-service-degrade-rules	DEFAULT_GROUP		<a href="#">详情</a>   <a href="#">示例代码</a>   <a href="#">编辑</a>   <a href="#">删除</a>   <a href="#">更多</a>

[https://blog.csdn.net/qq\\_27384769](https://blog.csdn.net/qq_27384769)

#### 3.1、例子：根据自定义API分组进行流控

gateway-service-sentinel-api

- **用户自定义的 API 定义分组**，定义一个 API 叫 **loit-portal-api**，请求 path 模式为/api-portal/\*\* 和 /loit-portal/\*\* 的都归到 my\_api 这个 API 分组下面。限流的时候可以针对这个自定义的 API 分组维度进行限流。
- 创建了针对该 API 分组的**单独流控规则**，允许**每个 URL** 的 QPS 限流
- 例子说明：对微服务**loit-portal**统一限流
- matchStrategy: 0 表示精确匹配。matchStrategy: 1 表示精确匹配。

```
[{
  "apiName": "loit-portal-api",
  "predicateItems": [
    {
      "pattern": "/api-portal/**",
      "matchStrategy": 1
    }, {
      "pattern": "/loit-portal/**",
      "matchStrategy": 1
    }, {
      "pattern": "/api-portal/api/v1/dict/list",
      "matchStrategy": 0
    }
  ]
}]
```



```
]
}
]
```

\* **Data ID:** gateway-service-sentinel-api


\* **Group:** DEFAULT\_GROUP

[更多高级选项](#)

**描述:** null

**Beta发布:** ☐ 默认不要勾选。

**配置格式:** ☐ TEXT ☒ JSON ☐ XML ☐ YAML ☐ HTML ☐ Properties

**配置内容** :

```
1  [{
2      "apiName": "loit-portal-api",
3      "predicateItems": [
4          {
5              "pattern": "/api-portal/**",
6              "matchStrategy": 1
7          }
8      ]
9  }
10 ]
11
```

[https://blog.csdn.net/qq\\_27384769](https://blog.csdn.net/qq_27384769)

gateway-service-sentinel-gateway

- 针对这个自定义的 API:**loit-portal-api** 分组维度进行限流
- 例子说明 api分组限流规则: 每秒钟 (intervalSec: 1) 限制次数 8 (count: 8)
- 其中限流参数及规则参照第一章中的“**网关限流参数组合组合说明**”

```
[ {
  "resource": "loit-portal-api",
  "count": 8,
  "intervalSec": 1
}
```

\* **Data ID:** gateway-service-sentinel-gateway

\* **Group:** DEFAULT\_GROUP

[更多高级选项](#)

**描述:** null

**Beta发布:** ☐ 默认不要勾选。

**配置格式:** ☐ TEXT ☒ JSON ☐ XML ☐ YAML ☐ HTML ☐ Properties

**配置内容 ? :**

```
1 [{
2   "resource": "loit-portal-api",
3   "count": 5000,
4   "intervalSec": 1
5 }
6 ]
```

[https://blog.csdn.net/qq\\_27384769](https://blog.csdn.net/qq_27384769)

### 3.2、例子：根据route\_id 进行限流

其中spring gateway配置的路由如下: 其中routeId为: **loit-portal-id**

```
spring:
  cloud:
    gateway:
      routes:
        - id: loit-portal-id
          uri: lb://loit-portal
          predicates:
            - Path=/api-portal/**
          filters:
            - StripPrefix=1
          logoutSign: true
```

gateway-service-sentinel-api

- 配置网关限流规则：每秒钟（intervalSec：1）限制次数 1（count：1）

```
[{
  "resource": "loit-portal-id",
  "count": 1,
  "intervalSec": 1
}]
```

### 3.3、例子：根据参数限流（待验证）

```
[
  {
    "resource": "cloud-discovery-client",
    "count": 10,
    "intervalSec": 2,
    "controlBehavior": 1,
    "maxQueueingTimeoutMs": 200,
    "paramItem": {
      "parseStrategy": 3,
      "fieldName": "test"
    }
  }
]
```

## 四、降级

注：待优化

降级关键配置

```
#spring.cloud.sentinel.datasource.ds2.file.file=classpath: degraderule.json
#spring.cloud.sentinel.datasource.ds2.file.data-type=json
#spring.cloud.sentinel.datasource.ds2.file.rule-type=degrade

spring:
  sentinel:
    datasource:

      ds3:
        nacos:
          server-addr: 47.114.50.99:8010
          namespace: 46c2400a-5773-4c26-be68-2e90a673259b
          dataId: ${spring.application.name}-degrade-rules
          data-type: json

          rule-type: degrade
```

```
#      datasource.ds3.file:
#      file: "classpath: gateway-service-degrade-rules"
#      ruleType: degrade
#      dataType: json

# 断路器设置
feign:
  sentinel:
    enabled: true
```

## 降级规则配置

```
[
  {
    "resource": "abc0",
    "count": 20.0,
    "grade": 0,
    "passCount": 0,
    "timeWindow": 10
  },
  {
    "resource": "abc1",
    "count": 15.0,
    "grade": 0,
    "passCount": 0,
    "timeWindow": 10
  }
]
```

DegradeRule

## 五、核心代码解读

---

### 1、Nacos的客户端从获取配置文件代码

```

private String getConfigInner(String tenant, String dataId, String group, long timeoutMs) throws NacosException {
    group = null2defaultGroup(group);
    ParamUtils.checkKeyParam(dataId, group);
    ConfigResponse cr = new ConfigResponse();

    cr.setDataId(dataId);
    cr.setTenant(tenant);
    cr.setGroup(group);

    // 优先使用本地配置
    String content = LocalConfigInfoProcessor.getFailover(agent.getName(), dataId, group, tenant);
    if (content != null) {
        // 本地配置不存在
        LOGGER.warn("{} [get-config] get failover ok, dataId={}, group={}, tenant={}, config={}", agent.getName(),
            dataId, group, tenant, ContentUtils.truncateContent(content));
        cr.setContent(content);
        configFilterChainManager.doFilter(request: null, cr);
        content = cr.getContent();
        return content;
    }

    try {
        content = worker.getServerConfig(dataId, group, tenant, timeoutMs);
        cr.setContent(content);
        configFilterChainManager.doFilter(request: null, cr);
        content = cr.getContent();
        return content;
    } catch (NacosException ioe) {
        // 服务端已经关闭, 进入catch
        if (NacosException.NO_RIGHT == ioe.getErrCode()) {
            throw ioe;
        }
        LOGGER.warn("{} [get-config] get from server error, dataId={}, group={}, tenant={}, msg={}",
            agent.getName(), dataId, group, tenant, ioe.toString());
    }

    LOGGER.warn("{} [get-config] get snapshot ok, dataId={}, group={}, tenant={}, config={}", agent.getName(),
        dataId, group, tenant, ContentUtils.truncateContent(content));
    content = LocalConfigInfoProcessor.getSnapshot(agent.getName(), dataId, group, tenant);
    cr.setContent(content);
    configFilterChainManager.doFilter(request: null, cr);
    content = cr.getContent();
    // 读取快照
    return content;
}

```

示例 使用http工具查看nacos上的配置文件

```

http://39.100.254.140:8103/nacos/v1/cs/configs?dataId=gateway-service-sentinel-
gateway&group=DEFAULT_GROUP&tenant=195cfbd3-bd2b-433b-b673-d440d4f8d234

```

## 六、待完善

- 1、系统保护规则 (LOAD、RT、线程数、入口QPS、CPU使用率)
- 2、授权规则: (白名单、黑名单)
- 3、集群流控
- 4、热点规则
- 5、dashboard配置存储到nacos