文章
Claire Zheng · 一月 20, 2021 阅读大约需 8 分钟

# InterSystems API 管理器简介

大家可能已经听说过，我们近期推出了InterSystems API管理器 **(以下简称IAM)。**InterSystems IRIS数据平台™新增了一项功能，支持用户监视、控制和管理IT基础架构中基于Web的API间通信。

在本文中，我将向大家展示如何设置IAM，并重点介绍IAM中可用的一些功能。InterSystems API管理器可提供你所需的一切功能。

- 监视基于HTTP的API通信，并了解谁在使用你的API、你最受欢迎的API是什么，哪些可能需要重新实现。
- 使用多种方式对API用户进行控制及限制。从简单的访问限制、API流量限制，到请求有效负载微调，你可以进行细粒度控制并快速做出反应。
- 使用集中式安全机制（如OAuth2.0或密钥和令牌身份验证）保护API。
- 招募第三方开发人员，为第三方开发人员提供一个专门的开发门户来满足他们的需求，并从一开始就为他们提供良好的开发体验。
- 扩展API需求并实现低延迟响应。

我很高兴为大家介绍IAM，让您一睹为快。

## 入门

从WRC Software Distribution站点下载IAM，并将其部署为自身的docker容器。

请确保满足以下最低要求：

- Docker引擎可用。最低支持版本是17.04.0+。
- Docker-compose CLI工具可用。最低支持版本是1.12.0+。

第一步需要加载docker镜像，通过如下：

```
docker load -i iam_image.tar
```

这样一来，IAM镜像可在你的计算机进行后续使用。IAM作为一个独立运行的容器，可以单独从InterSystems IRIS后端对其进行扩展。

启动IAM前，需要访问IRIS实例来加载所需的许可证信息。须进行以下配置更改：

- 启用/api/IAM web应用程序
- 启用IAM用户
- 更改IAM用户密码

现在，我们可以开始配置IAM容器了。在distribution tarball里可以找到一个名为 “ iam-setup ” 的Windows和Unix系统脚本。该脚本可帮助你正确地设置环境变量，使IAM容器能够与InterSystems IRIS实例建立连接。这是我在Mac终端会话中的运行示例：

```
source ./iam-setup.sh
Welcome to the InterSystems IRIS and InterSystems API Manager (IAM) setup script.
This script sets the ISC_IRIS_URL environment variable that is used by the IAM contai
ner to get the IAM license key from InterSystems IRIS.
Enter the fu
ll image repository, nam
e and tag for your IAM docker image: intersystems/iam:0.34-1-1
Enter the IP address for your InterSystems IRIS instance. The IP address has to be ac
cessible from within the IAM container, therefore, do not use "localhost" or "127.0.0
.1" if IRIS is running on your local machine. Instead use the public IP address of yo
ur local machine. If IRIS is running in a container, use the public IP address of the
 host environment, not t
he IP address of the IRIS container. xxx.xxx.xxx.xxx
Enter the web server port for your InterSystems IRIS instance: 52773
Enter the password for the IAM user for your InterSystems IRIS instance:
Re-enter your password:
Your inputs are:
Full image repository, name and tag for your IAM docker image: intersystems/iam:0.34-
1-1
IP address for your InterSystems IRIS instance: xxx.xxx.xxx.xxx
Web server port for your InterSystems IRIS instance: 52773
Would you like to continue with these inputs (y/n)? y
Getting IAM license using your inputs...
Successfully got IAM license!
The ISC_IRIS_URL environment variab
le was set to: http://IAM:****************@xxx.xxx.xxx.xxx:52773/api/iam/license
WARNING: The environment variable is set for this shell only!
To start the services, run the following command in the top level directory: docker-
compose up -d
To stop the services, run the following command in the top level directory: docker-
compose down
URL for the IAM Manager portal: http://localhost:8002
```

我隐藏了IP地址和密码，但这足以让大家了解配置是多么简单。现在我们得到了开始下一步前所需的全部内容：InterSystems IRIS实例的完整镜像名称、IP地址和端口，以及IAM用户密码。

现在可以通过执行以下命令启动IAM容器：

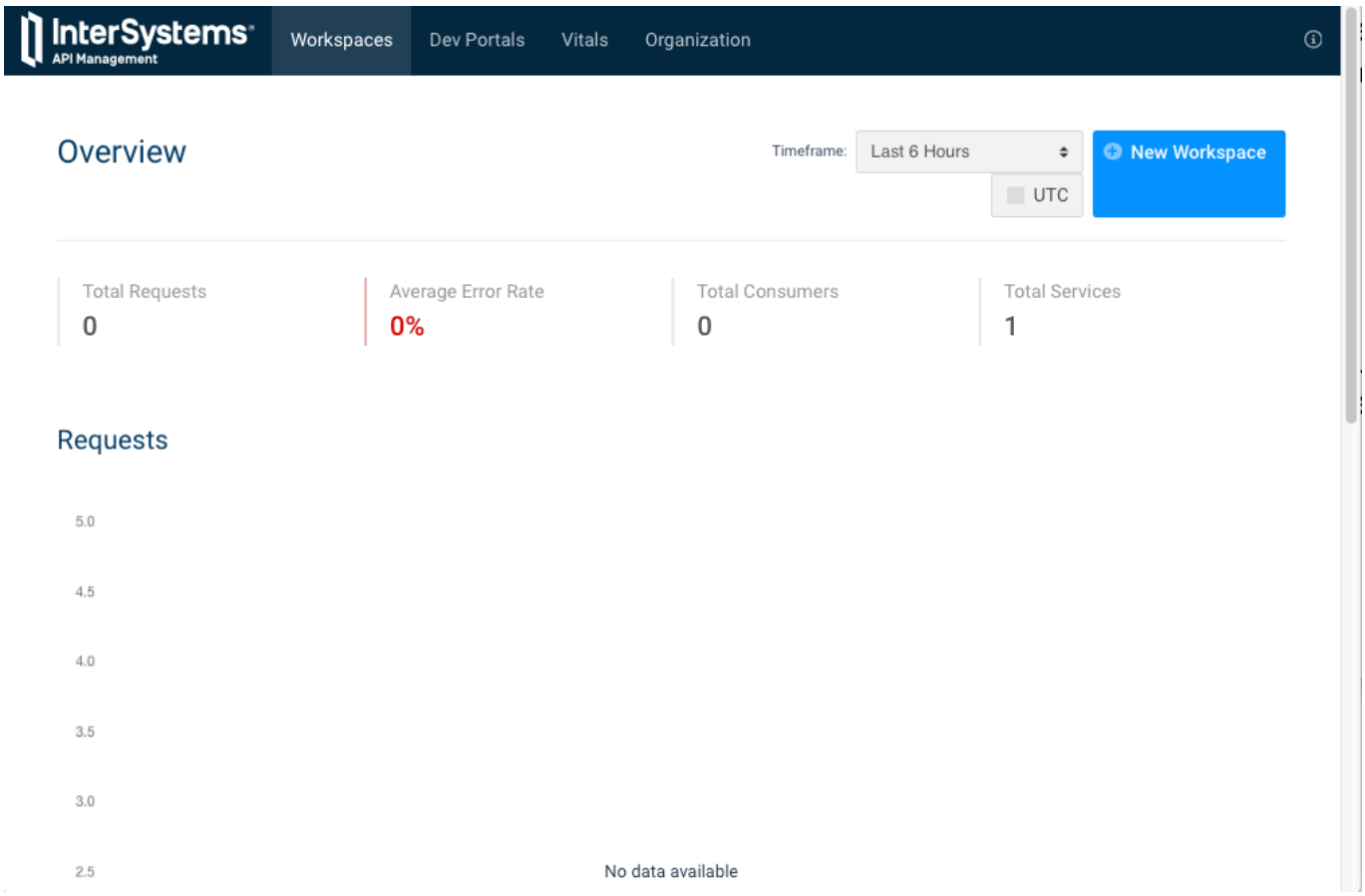
```
docker-compose up -d
```

该命令将开始协调IAM容器，并确保以正确的顺序启动所有内容。

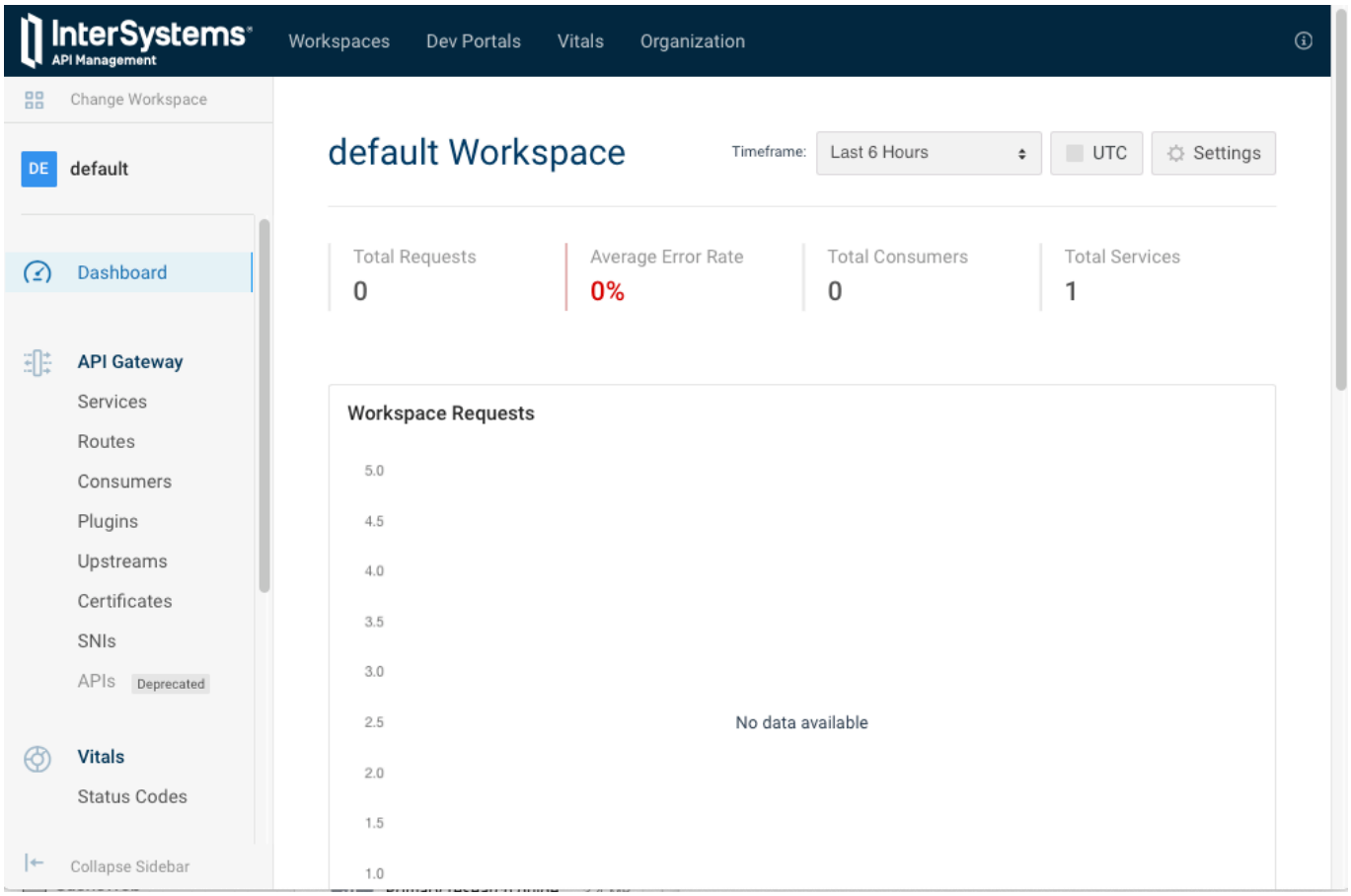
你可以使用以下命令检查容器的状态：

```
docker ps
```

在浏览器中输入**localhost:8002**会出现基于web的用户界面：




因为这是一个全新的节点，所以全局报告中未显示任何吞吐量。但我们很快就会改变这个情况。我们可以看到，IAM 支持 “ Workspace（工作区） ” 概念，将工作划分为 “ module ” 和/或 “ team ” 。向下滚动并选择 “ default ” 工作区会将我们带到 “ Dashboard ” 界面。我们将在 “ default ” 工作区开始首次实验。



同样，这个工作区的请求数量也是零，但是你可以先在左侧的菜单中了解一下API网关的重要概念。前两个元素——即服务和路由——是最重要的。服务是指向用户公开的API。因此，IRIS实例中的REST API被视为一种服务，就像你所使用的Google API一样。路由决定应将传入请求路由到哪些服务。每个路由都有一组特定的条件，如果满足这些条件，就会将请求路由到相关的服务。大家需要了解的是，路由可以匹配发送者的IP或域、HTTP方法、部分URI，或者其中的几种。

现在让我们创建一个IRIS实例的服务，其值如下：

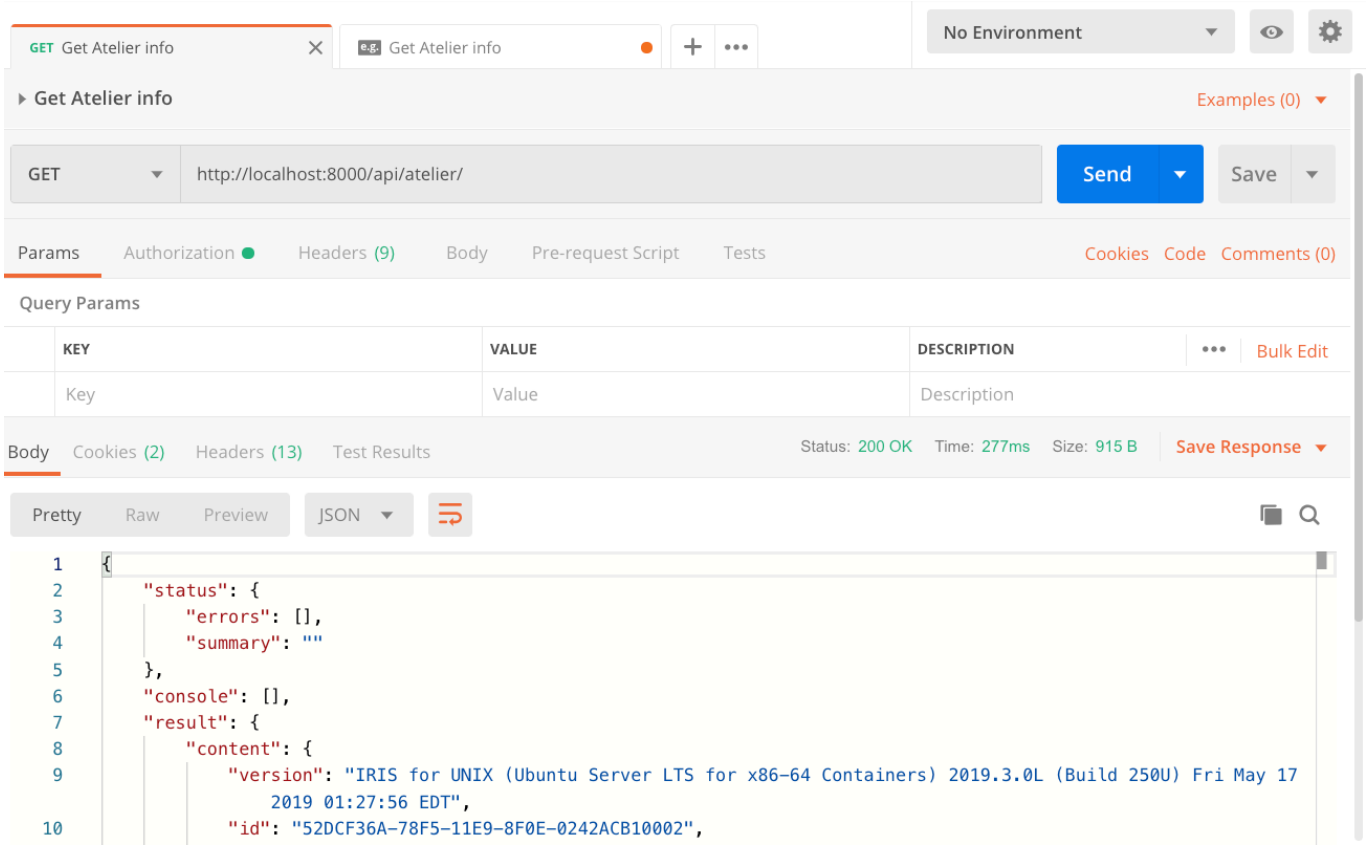
| 字段 | 值 | 说明 |
|---|---|---|
| name | test-iris | 此服务的逻辑名称 |
| host | xxx.xxx.xxx.xxx | IRIS实例的主机公开IP地址 |
| port | 52773 | 用于HTTP请求的端口 |
| protocol | http | 要支持的协议 |

保留其他所有内容的默认设置。现在让我们创建一个路由：

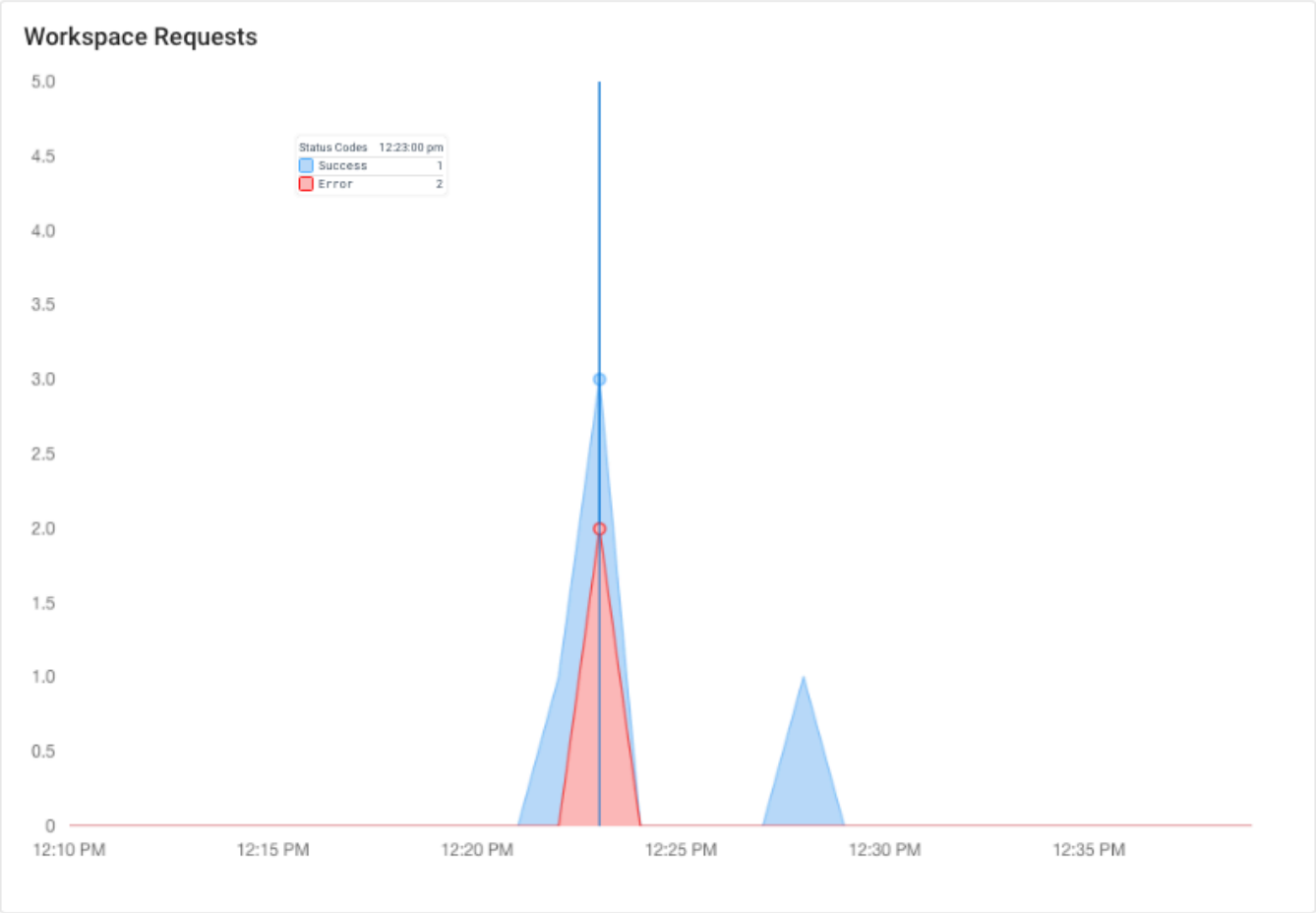
| 字段 | 值 | 说明 |
|---|---|---|
| paths | / api /atelier | 使用此路径的请求将转发到我们的IRIS实例 |
| protocols | http | 你希望支持的协议 |
| service | test-iris | 与此路由匹配的请求将被转发到此服务 |

同样，对其他所有内容保留默认设置。默认情况下，IAM正在监听端口8000上的传入请求。从现在开始，发送到http://localhost:8000并以/api/atelier路径开头的请求将被路由到IRIS实例。

让我们在REST客户端尝试一下（我使用的是Postman）。




向http://localhost:8000/api/atelier/
发送一个GET请求会从IRIS实例返回一个响应。每个请求都经过IAM，并监视HTTP状态码、延迟和用户（如果已配置）等指标。我接着发出了另外几个请求（包含对不存在的端点的两个请求，如/api/atelier/test/），可以在Dashboard中看到汇总：




## 使用插件

既然已经有了一个基本的路由，那么可以开始管理API流量了。现在我们可以开始添加补充服务的行为。现在可以创造奇迹了。

执行某种行为最常见的方法就是添加插件。插件可提供一些功能，并且通常可以附加到IAM的某个部分。它们可能会对全局运行产生影响，也可能只对单个用户（组）、服务或路由等产生影响。首先，我们在路由中添加限速插件。此时需要确保插件和路由之间建立的链接是路由的唯一ID。这些可以从路由的详细信息里找到。

# Viewing Route

| | |
|---|---|
| **created_at** | June 24th 2019, 3:56:46pm |
| **hosts** | |
| **id** | d6a97e5b-7da6-4c98-bc69-6a09263039a8 |
| **methods** | |
| **paths** | /api/atelier |
| **preserve_host** | false |
| **protocols** | http, https |
| **regex_priority** | 0 |
| **service** | test-iris |
| **strip_path** | false |
| **updated_at** | June 24th 2019, 3:56:46pm |

如果按照本文的步骤进行，那么你的路由ID是不一样的。复制ID继续下一步。

单击左侧工具栏菜单上的 **“ Plugins ”**
。通常可以在此界面上看到所有的活动插件，但由于这个节点相对较新，所以未显示任何活动插件。选择 **“ Add New Plugin ”**继续下一步。

我们要选择的插件在 “ Traffic Control ” 类中，名为 “ Rate
Limiting ” 。选中该插件。由于插件非常灵活，所以我们可以在这里定义非常多的字段，但现在我们只关心两个字段：

| 字段 | 值 | 说明 |
|---|---|---|
| route_id | d6a97e5b-7da6-4c98-bc69-6a09263039a8 | 将路由ID粘贴到此处 |
| config.minute | 5 | 每分钟允许调用的次数 |

如上所示，插件已配置并处于活动状态。你可能已经发现有多种时间间隔可以选择，比如分钟、小时或天。我特意选择了分钟，因为这样可以让我们很容易理解这个插件产生的效果。

如果在Postman中再次发送相同的请求，**会发现响应返回了两个附加的头信息**：XRateLimit-Limit-minute (value 5)

和XRateLimit-Remaining-minute(value
4)。这是在告诉客户端，每分钟最多可以调用5次，并且在当前时间间隔内还有4个请求可用。

Body Cookies (2) Headers (15) Test Results Status: 200 OK Time: 289ms

| KEY | VALUE |
|---|---|
| Content-Type | application/json; charset=utf-8 |
| Content-Length | 388 |
| Connection | keep-alive |
| X-RateLimit-Limit-minute | 5 |
| X-RateLimit-Remaining-minute | 4 |

如果不断地发出相同的请求，最终会用完可用配额，得到一个带有以下负载的**HTTP状态码429：**

Body Cookies (2) Headers (7) Test Results Status: 429 Too Many Requests Time: 9ms

Pretty Raw Preview JSON

```
1 {
2     "message": "API rate limit exceeded"
3 }
```

等这一分钟结束后，就可以再次调用。这是一个非常方便的机制，可以完成以下工作：

1. 确保后端避免高峰值
2. 为客户端设置一个期望值，即允许以透明的方式为服务进行多少次调用
3. 引入分级制有望从API流量中获利（例如，青铜级别每小时调用100次，而黄金级别则不受限制）

你可以为不同的时间间隔设置值，从而在一定时期内平滑API流量。假设允许某条路由每小时进行600次调用，平均每分钟调用10次。但是，你没有阻止客户端在一小时的第一分钟就用光600次调用（也许这就是你想要的）。也行你想让负载在一个小时内分配得更均匀。将**configminute字段**
设置为20，这样就可以确保用户每分钟调用的次数不超过20次，每小时不超过600次。这将使分钟级别的间隔出现一些峰值，因为它们平均每分钟只能调用10次，但用户不能在一分钟内用完一小时配额。现在，至少需要30分钟，系统才会达到满负荷运行。客户端将在每个配置的时间间隔内收到附加标头，例如：

| header | value |
|---|---|
| X-RateLimit-Limit-hour | 600 |
| X-RateLimit-Remaining-hour | 595 |
| X-RateLimit-Limit-minute | 20 |
| X-RateLimit-Remaining-minute | 16 |

当然，可以采用多种不同的方法配置rate-limits，这取决于你想要实现的目标。

对此我不再做过多介绍，因为作为一篇介绍InterSystems API Manager的文章，上述介绍已经足够了。
**IAM还可以用来实现更多的事情。我们刚刚只用了40多个插件中的一个，**甚至还没有使用到所有的核心概念，你还可以实现以下任务：

- 为所有服务添加集中式身份验证机制
- 通过负载均衡请求扩展到支持同一组API的多个目标
- 向更小的受众介绍新特性或bugfixes，并在向大家发布之前监视的进展情况
- 为内部和外部开发人员提供一个专用的、可自定义的开发人员门户，记录他们有权访问的所有API
- 缓存常见的请求响应，以减少响应延迟和服务系统上的负载

所以，请大家试一试IAM，并在下面评论区留下建议。我们努力推出这一功能，希望了解大家可以使用这项技术克服哪些挑战。

## 更多资源

官方新闻稿：InterSystems IRIS Data Platform 2019.2 introduces API Management capabilities

短动画视频概述：什么是InterSystems API Manager？（英文） What is InterSystems API Manager

8分钟小视频带你了解主要亮点：InterSystems API管理器简介（英文） Introducing InterSystems API Manager

选自IRIS文档部分内容：InterSystems API管理器文档（英文） InterSystems API Manager Documentation

**注：本文为译文，欢迎**点击查看原文**，原文由**Stefan Wittmann**撰写**

#API #InterSystems API管理器（IAM） #REST API #SOAP #InterSystems IRIS

---

**源URL:**https://cn.community.intersystems.com/post/intersystems-api-%E7%AE%A1%E7%90%86%E5%99%A8%E7%AE%80%E4%BB%8B