配置Postman以在vManage上执行API

目录

简介

本文档介绍如何使用Postman执行应用程序编程接口(API)。

系统要求

- Postman安装
- 访问vManage以及用户名和密码凭证

注意：如果您没有Postman，请从https://www.postman.com/downloads/下载它

背景信息

主要或最常用的HTTP动词（或方法，它们被正确称为）是POST、GET、PUT、PATCH和DELETE。

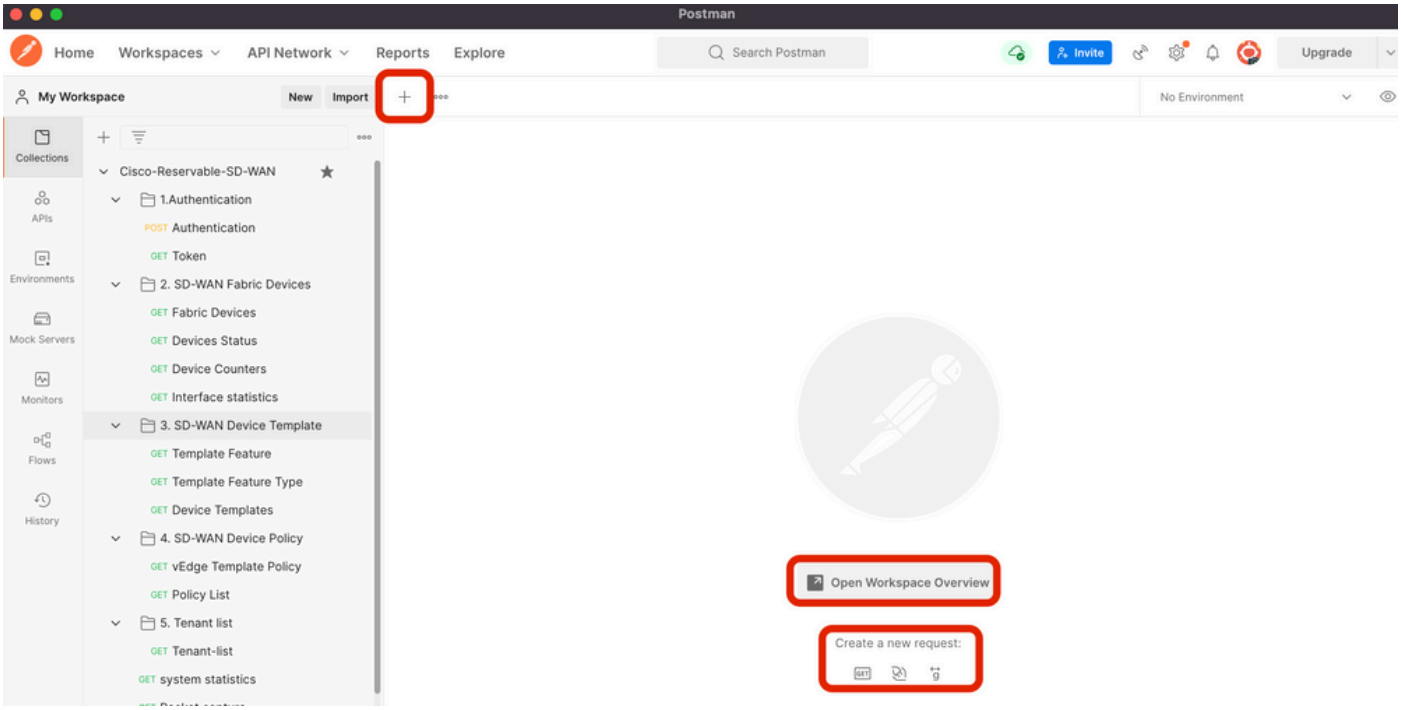
它们分别对应于创建、读取、更新和删除（或CRUD）操作。

也有许多其它动词，但使用频率较低。在这些频率较低的方法中，OPTIONS和HEAD的使用频率高于其他方法。

配置Postman以执行API

步骤1:打开Postman并创建新的HTTP请求。

如果单击任何突出显示的选项，则可以创建新的HTTP请求。



创建新的HTTP请求。

第二步：使用您的用户名和密码凭证验证到vManage。

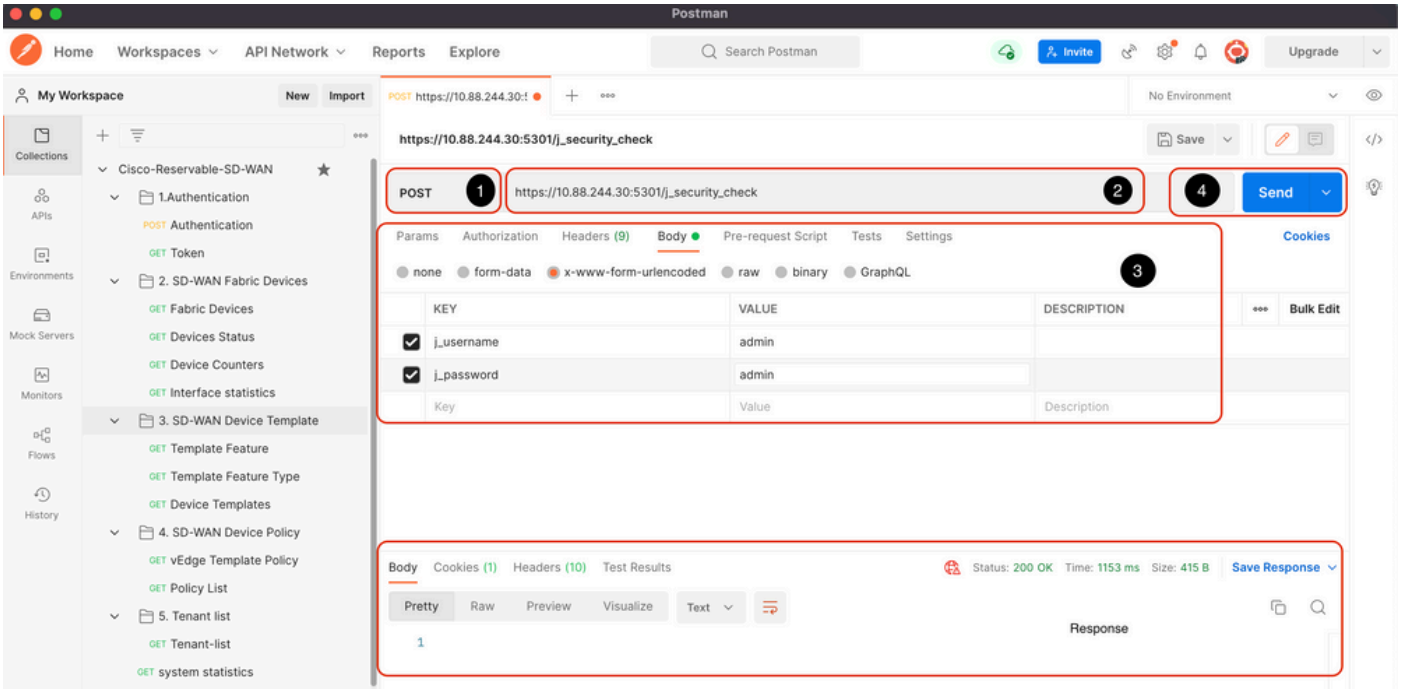
创建另一个HTTP请求。

1. 选择**POST**作为HTTP谓词。
2. 在POST旁添加https://<vmanage-ip>/j_security_checkbox。
3. 单击**Body**并分别添加为**KEY**参数**j_username**和**j_password**及其值。
4. 单击发送。

注意：在本示例中，vManage ip address为10.88.244.30，端口为5301

注意：用于用户名和密码值，我们使用admin。

在Postman中填写参数。



vManage authentication。

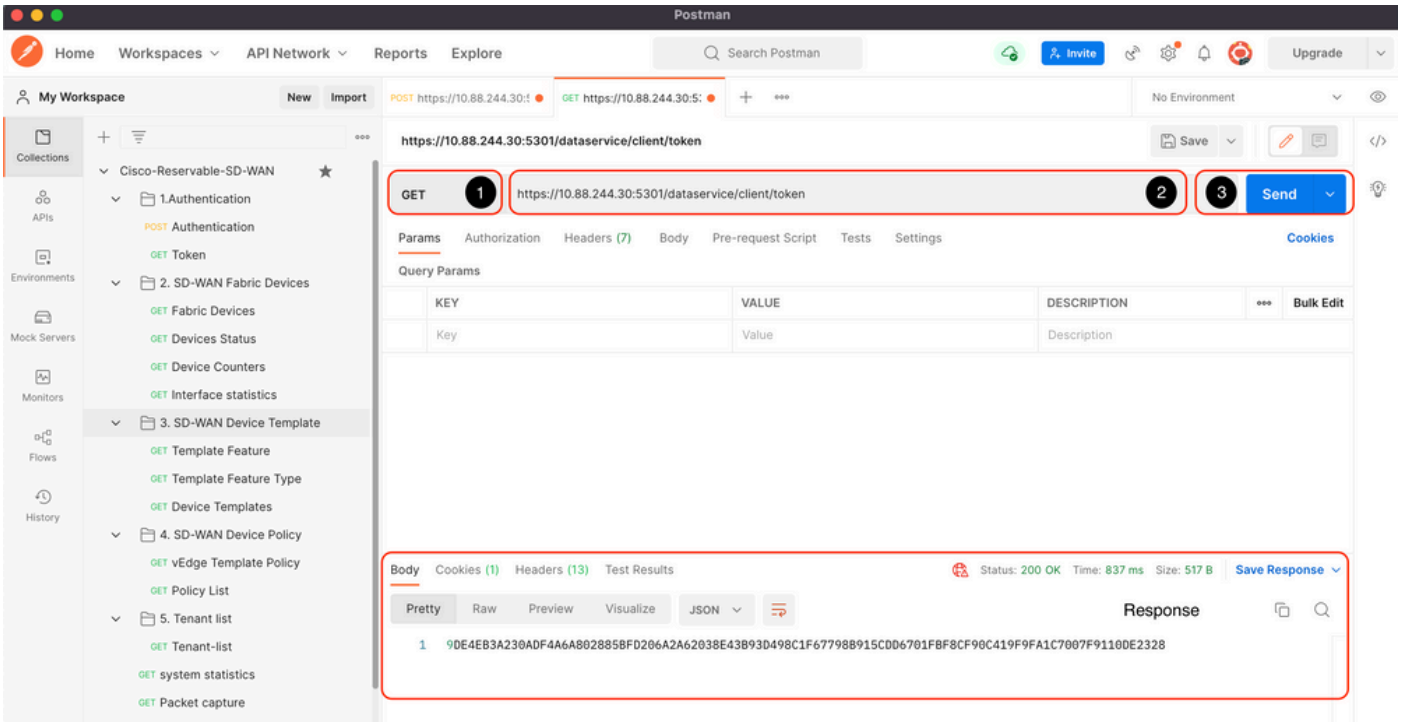
注意：此API调用的响应必须为空

第三步：请求令牌

1. 选择**GET**作为HTTP谓词。
2. 在GET https://<vmanage-ip>/dataservice/client/token旁边添加API调用详细信息
3. 单击**Send**

注：自vManage 19.2.1版起，成功登录的用户必须通过API调用为每个POST/PUT/DELETE操作发送X-XSRG令牌或CSRF令牌。

执行API调用后，您将在正文中获取响应字符串。保存该字符串。图中所示为Postman输出示例。



警告：如果您没有获得如图所示的令牌，请重复此步骤。

第四步：继续执行另一个用于vManage的API。

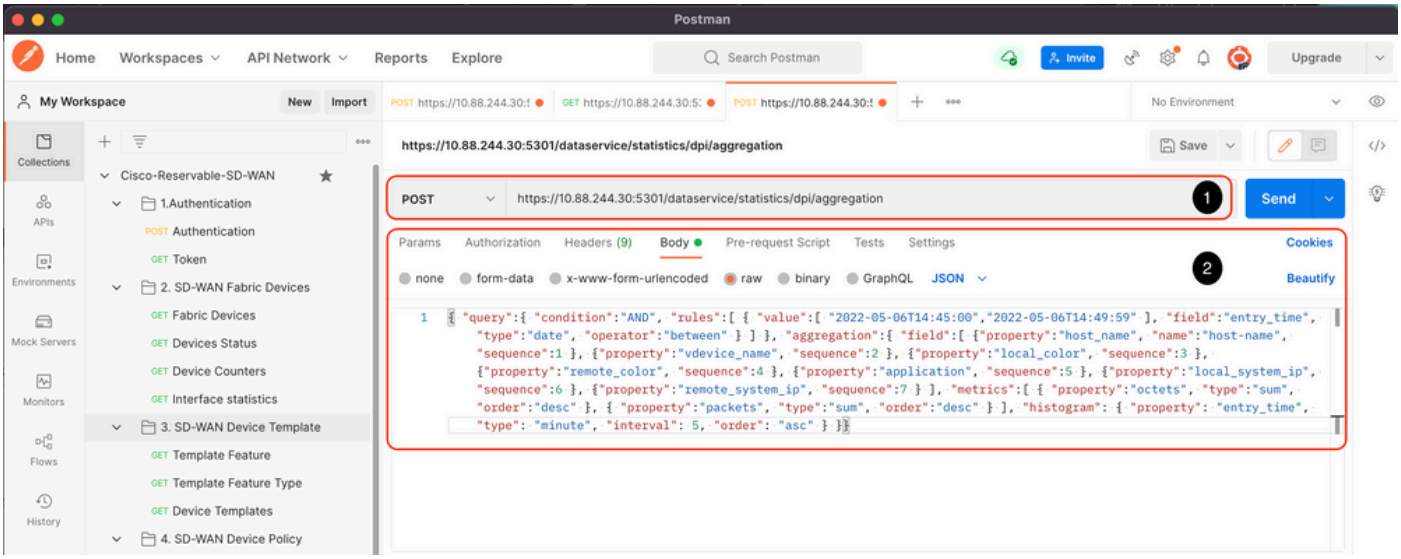
此示例包含POST请求

1. 选择要执行的API调用，在本例中为**https:///dataservice/statistics/dpi/aggregation**

   提示：如果您希望探索其他API调用，请转到vManage url https://vmanage-ip:port/apidocs

2.收集您的API调用正文。

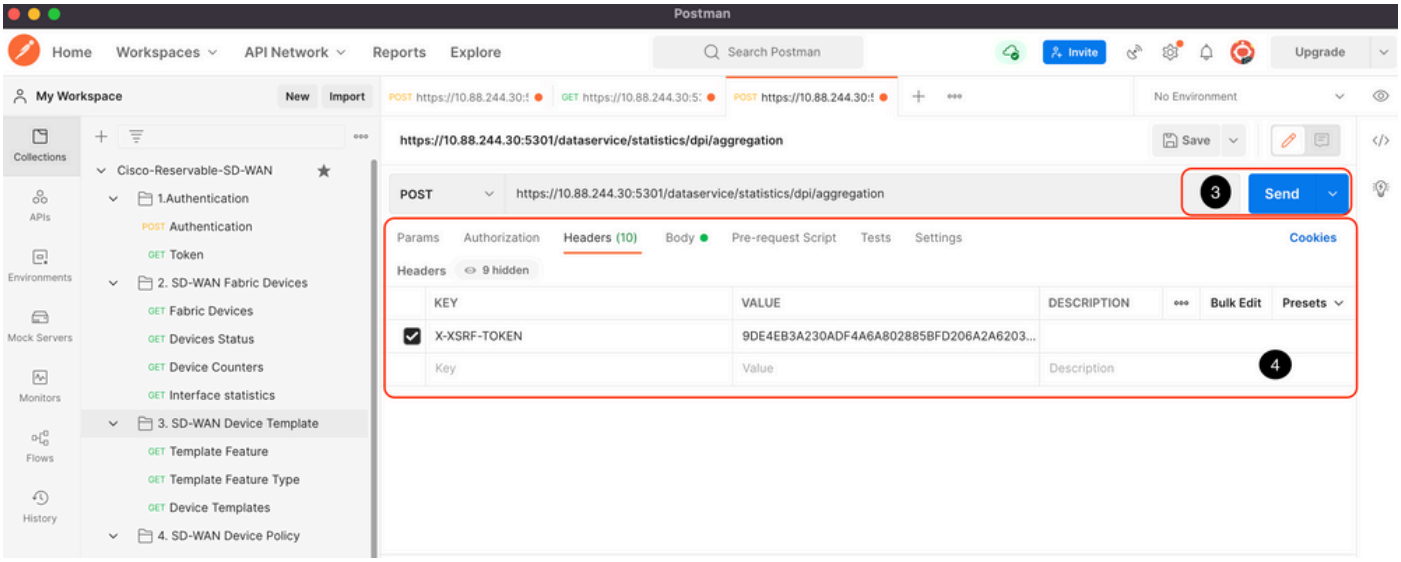
注意：此API调用包含JSON格式的正文



3.单击**Header**并添加字符串**X-XSRF-TOKEN**作为值，作为**Key**。

4.单击**发送**。

显示的图像显示了您的API调用必须如何显示。



DPI汇聚API调用。

第五步：关闭会话

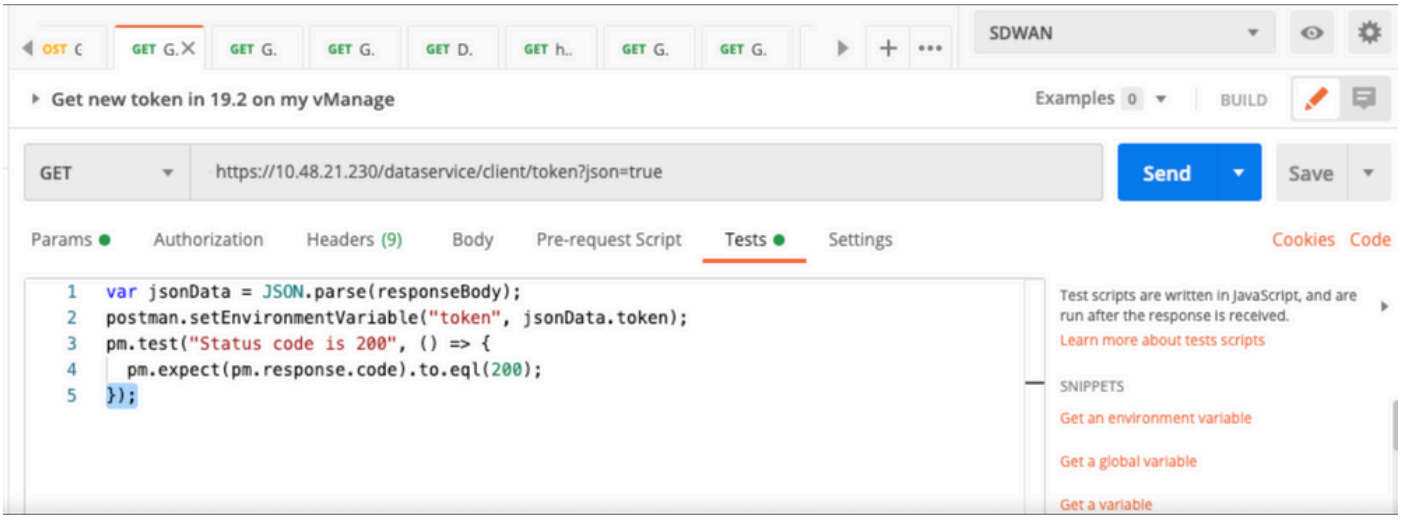
从vManage和/或设备检索所需的所有信息后，即可释放vManage的资源，消除恶意用户使用会话的可能性。

在自动化环境中运行API调用

保存Cookie和变量，以便在后续API调用中使用

如何在变量中保存令牌？

将令牌保存在变量中，以便以后重复使用。

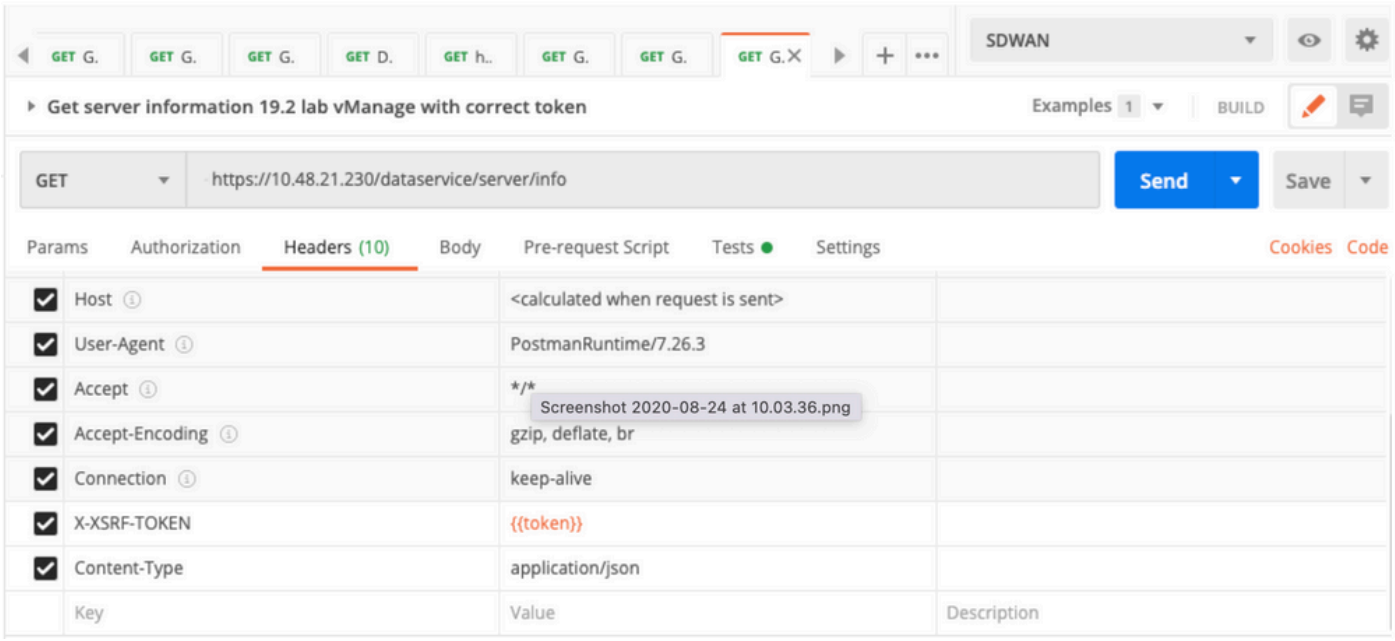


将令牌保存在变量中

当我们以JSON格式请求令牌时，请将其存储。使用**测试**选项卡并粘贴显示的行。

```
var jsonData = JSON.parse(responseBody);
postman.setEnvironmentVariable("token", jsonData.token);
```

之后，任何API调用都可以使用令牌变量。

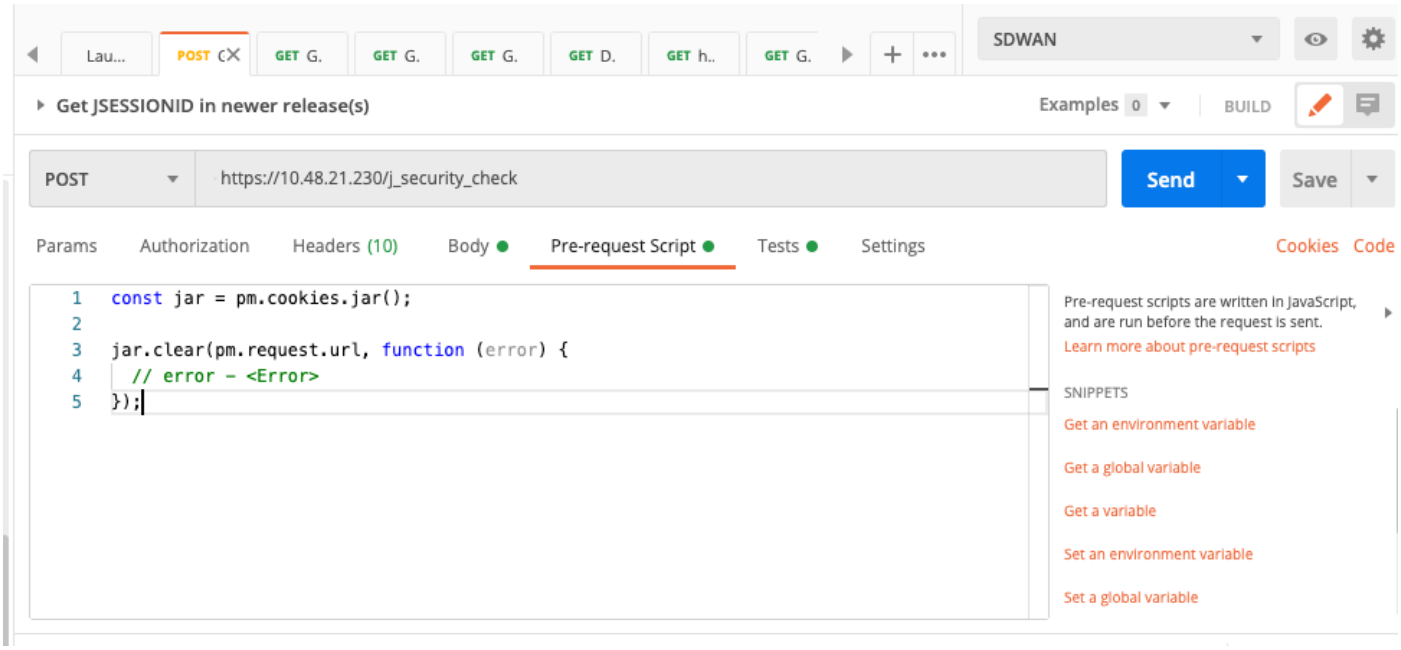


使用令牌变量

如何清除新会话的SESSIONID cookie?

每当执行API调用以退出时，请使用JSESSIONID。

我们不能像在早期版本中那样使用任何基本身份验证。相反，我们仅提供凭证并将ID保存在cookie中。在此之前，我们可以使用预测试来清除所有或特定的cookie。



清除Cookie

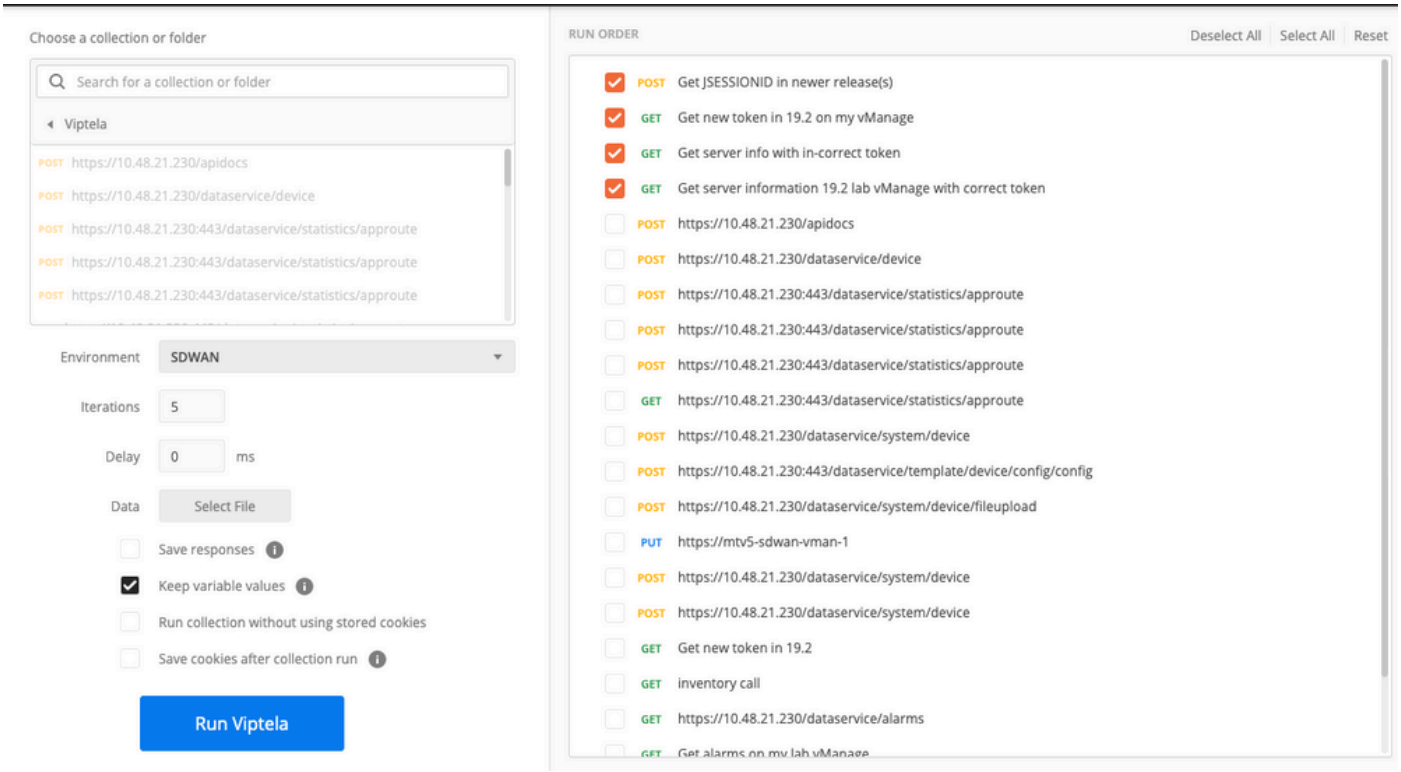
这是通过预请求脚本中的代码实现的。

如何使用Collection Runner

现在，我们拥有一些环境，可以在其中运行会话并保存特定于每个会话的数据，因此您可以使用“收集运行程序”运行一系列呼叫。

选择要重复的事件的顺序，选择重复计数，以便Postman可以执行API调用，这是选定的每次运行具

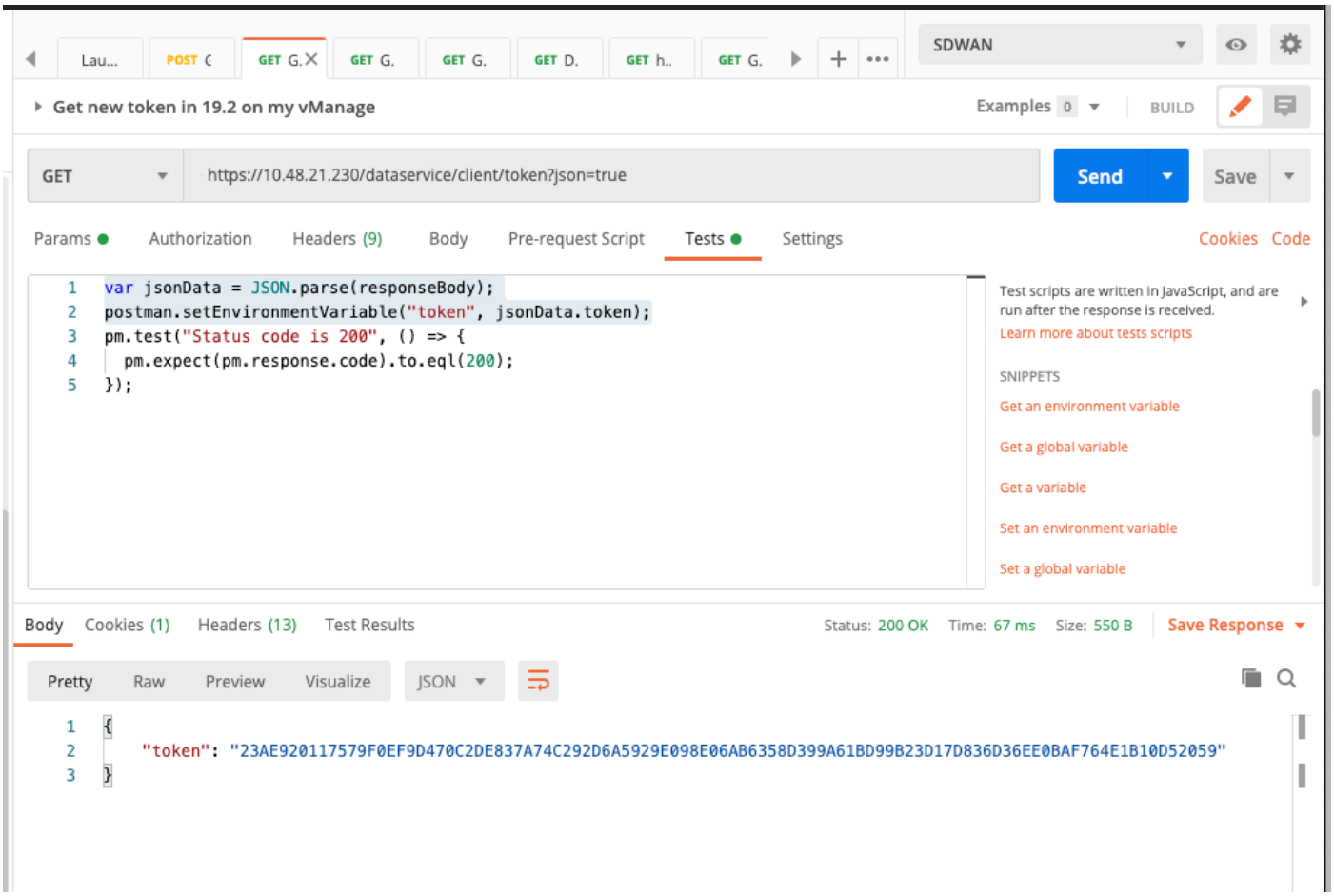
有结果的次数。



收集运行器

从调用的“库”中，按照一定的顺序放置它们，以获得要执行的特定流/命令。

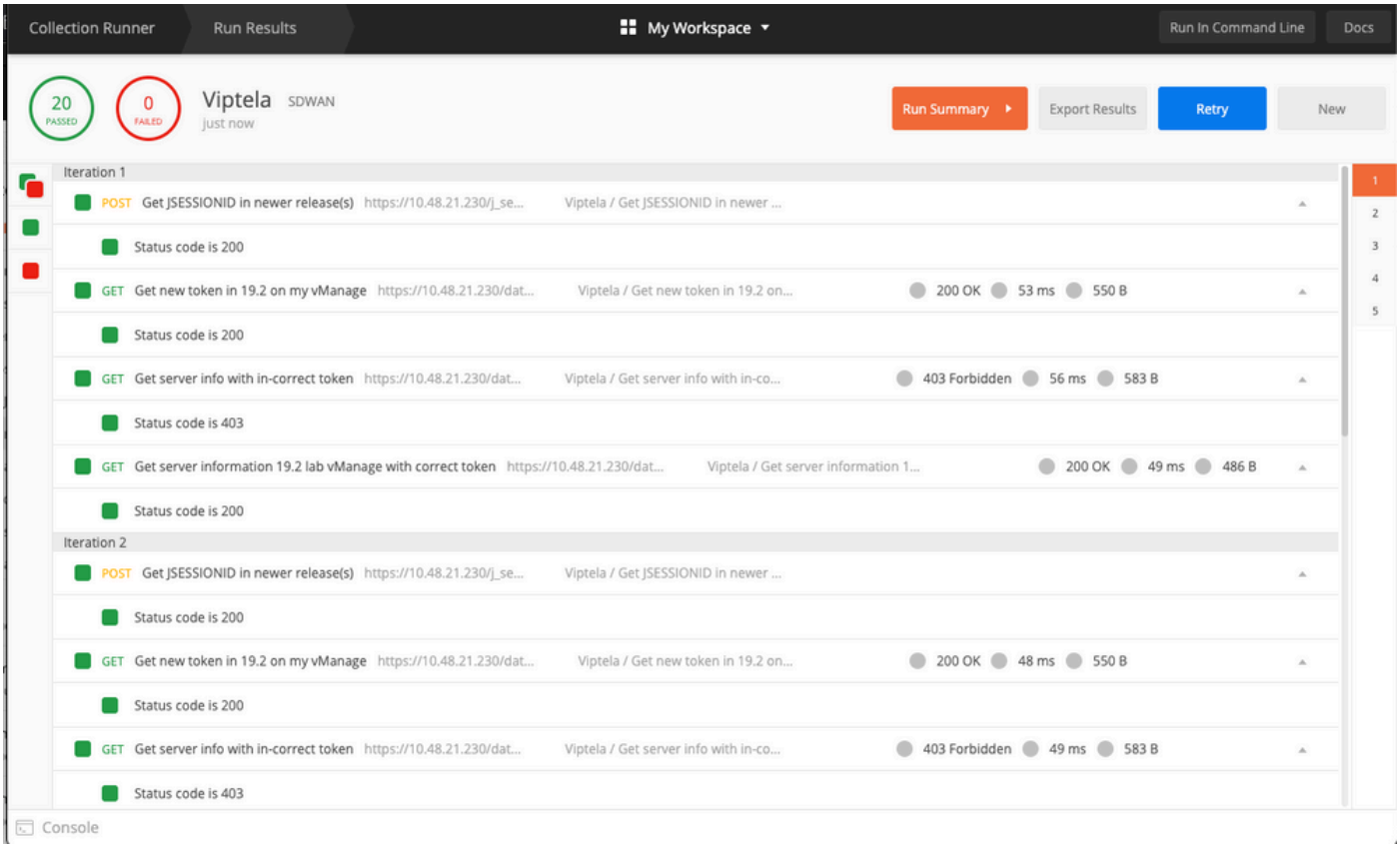
输入结果检查您是否得到200 OK或其他值作为响应，并将其视为通过或失败。



检查响应代码

```
pm.test("Status code is 200", () => {
 pm.expect(pm.response.code).to.eql(200);
});
```

然后我们就可以看到我们跑步的时候是通过了还是失败了。



自动运行

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。