

Configuración de Postman para ejecutar API en vManage

Contenido

[Introducción](#)

[Requisitos del sistema](#)

[Antecedentes](#)

[Configuración de Postman para ejecutar las API](#)

[Paso 1. Abra Postman y cree una nueva solicitud HTTP.](#)

[Paso 2. Realice la autenticación con sus credenciales de nombre de usuario y contraseña en vManage.](#)

[Paso 3. Solicitar un token](#)

[Paso 4. Proceda a ejecutar otra API en vManage.](#)

[Paso 5. Cierre de la sesión](#)

[Ejecución de llamadas API en un entorno automatizado](#)

[¿Cómo guardar el token en una variable?](#)

[¿Cómo se borra la cookie SESSIONID para las nuevas sesiones?](#)

[Cómo utilizar Collection Runner](#)

Introducción

Este documento describe cómo ejecutar interfaces de programación de aplicaciones (API) con Postman.

Requisitos del sistema

- Cartero instalado
- Acceso a vManage y credenciales de nombre de usuario y contraseña

Nota: Si no tiene Postman, descárguelo de <https://www.postman.com/downloads/>

Antecedentes

Los verbos HTTP primarios o los más utilizados (o métodos, como se les denomina correctamente) son POST, GET, PUT, PATCH y DELETE.

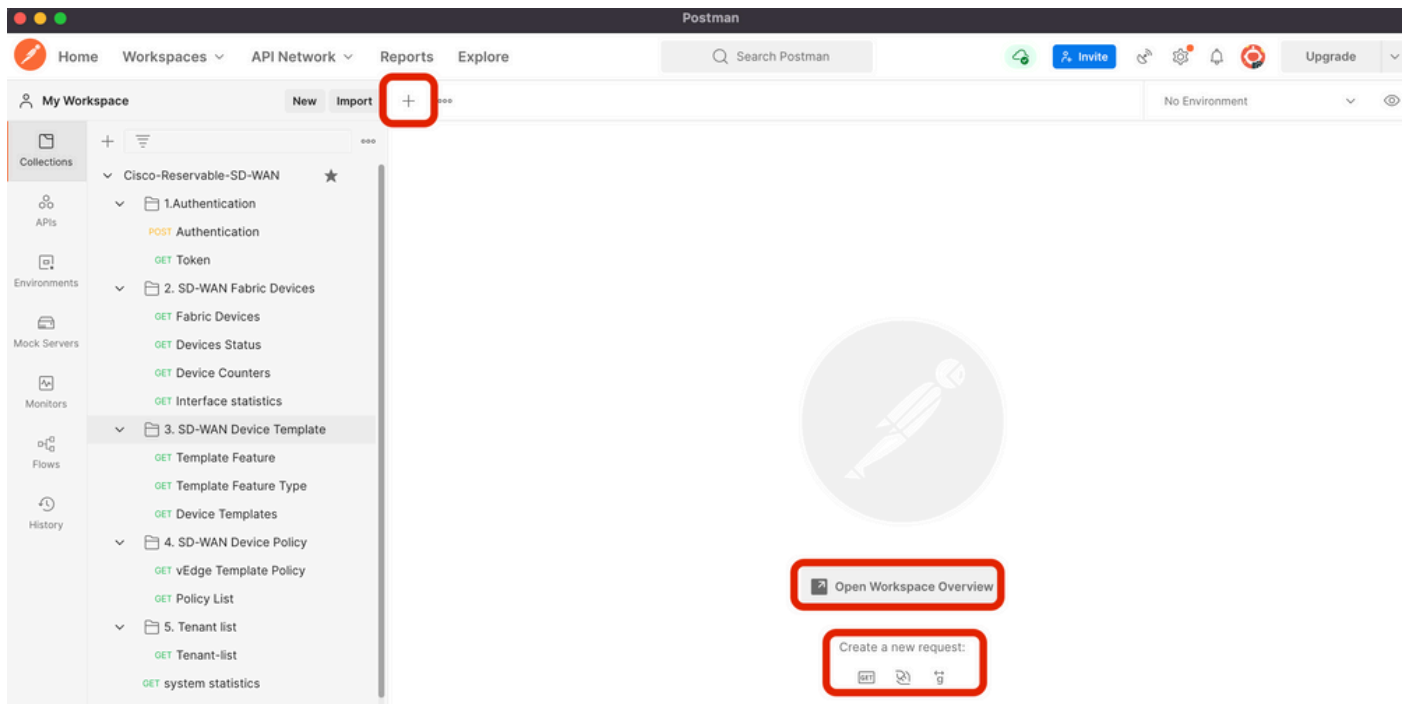
Corresponden a operaciones de creación, lectura, actualización y eliminación (o CRUD), respectivamente.

Hay una serie de otros verbos, también, pero se utilizan con menos frecuencia. De estos métodos menos frecuentes, OPTIONS y HEAD se utilizan con más frecuencia que otros.

Configuración de Postman para ejecutar las API

Paso 1. Abra Postman y cree una nueva solicitud HTTP.

Puede crear nuevas solicitudes HTTP si hace clic en cualquiera de las opciones resaltadas.



Cree una nueva solicitud HTTP.

Paso 2. Realice la autenticación con sus credenciales de nombre de usuario y contraseña en vManage.

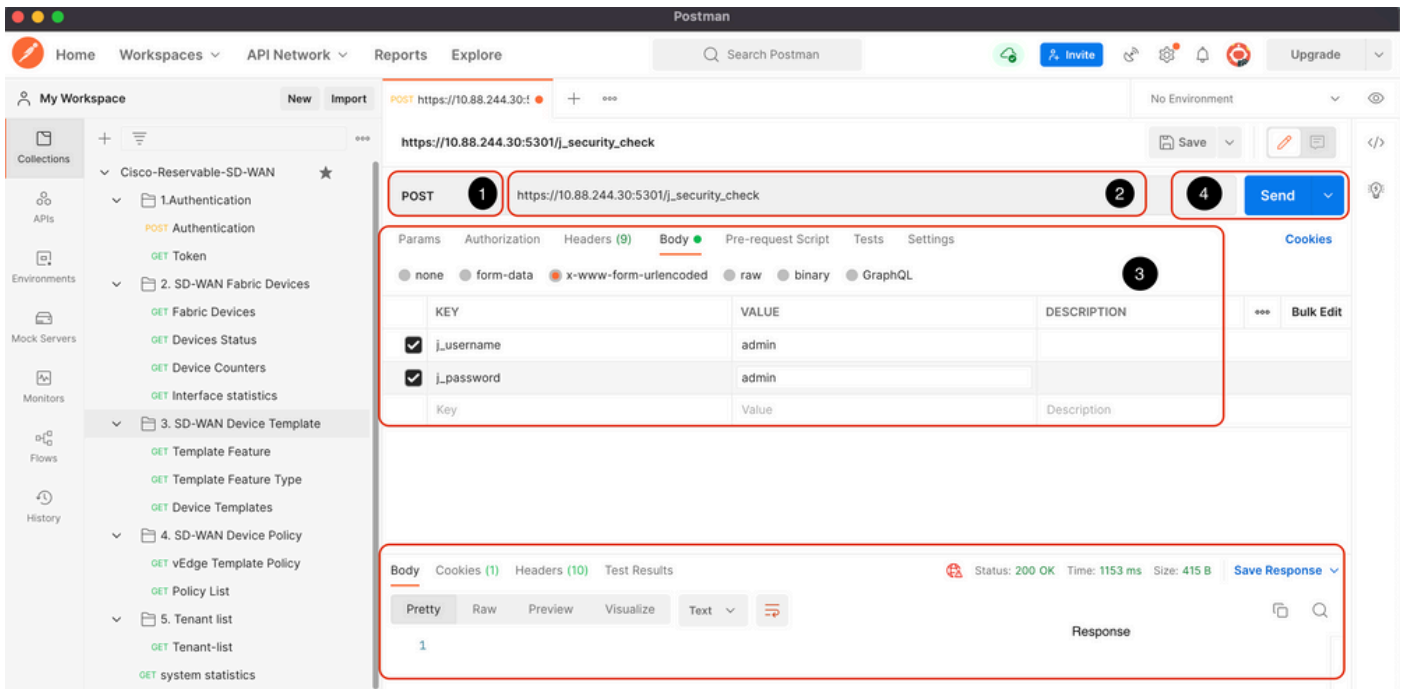
Cree otra solicitud HTTP.

1. Seleccione **POST** como verbo HTTP.
2. Agregue https://<vmanage-ip>/j_security_check a POST.
3. Haga clic en **Body** y agregue como **KEY** parámetros **j_username** y **j_password** y sus valores respectivamente.
4. Haga clic en **Enviar**.

Nota: En este ejemplo, la dirección IP de vManage es 10.88.244.30 y el puerto es 5301

Nota: Como valores de nombre de usuario y contraseña, utilizamos admin.

Cumplimente los parámetros en Postman.



Autenticación vManage.

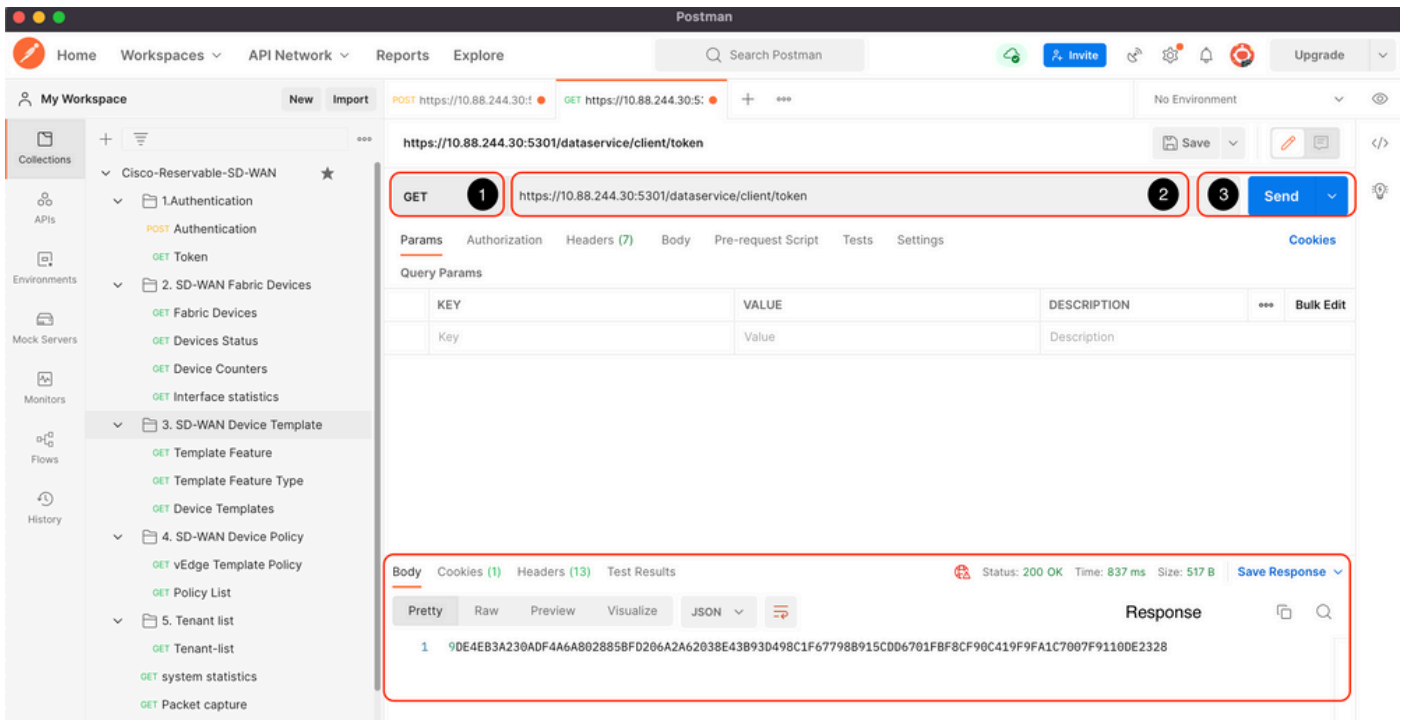
Precaución: la respuesta de esta llamada de API debe estar vacía

Paso 3. Solicitar un token

1. Seleccione **GET** como verbo HTTP.
2. Agregue los detalles de la llamada API junto a GET <https://<vmanage-ip>/dataservice/client/token>
3. Haga clic en **Enviar**

Nota: Desde la versión 19.2.1 de vManage, se ha hecho obligatorio que un usuario que haya iniciado sesión correctamente envíe el token X-XSRG-TOKEN o CSRF para cada operación POST/PUT/DELETE a través de una llamada API.

Una vez que se ejecuta la llamada API, se obtiene una cadena de respuesta en el cuerpo. Guarde esa cadena. La imagen que se muestra ejemplifica la salida In Postman.



Solicitar un token para vManage

Advertencia: Si no ha obtenido un token como se muestra en la imagen, repita el paso.

Paso 4. Proceda a ejecutar otra API en vManage.

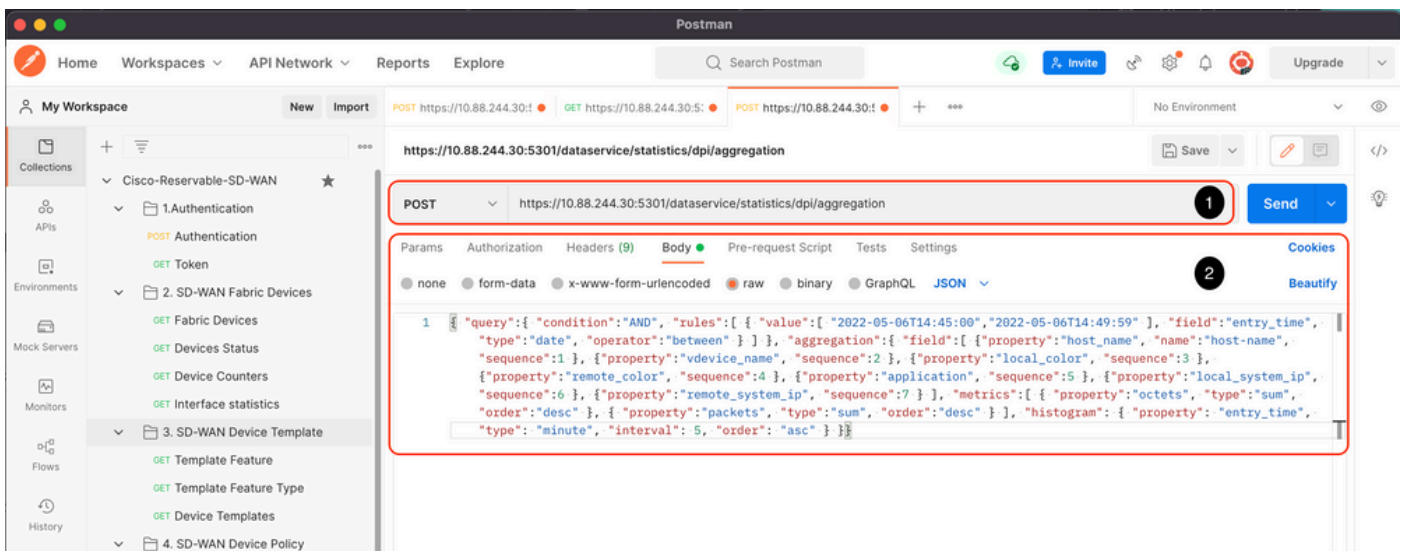
Este ejemplo implica una solicitud POST

1. Seleccione la llamada API que desea ejecutar; en nuestro caso, es <https://dataservice/statistics/dpi/aggregation>

Consejo: Si desea explorar otras llamadas de API, vaya a la URL de vManage <https://vmanage-ip:port/apidocs>

2. Recopile el cuerpo de llamadas de la API.

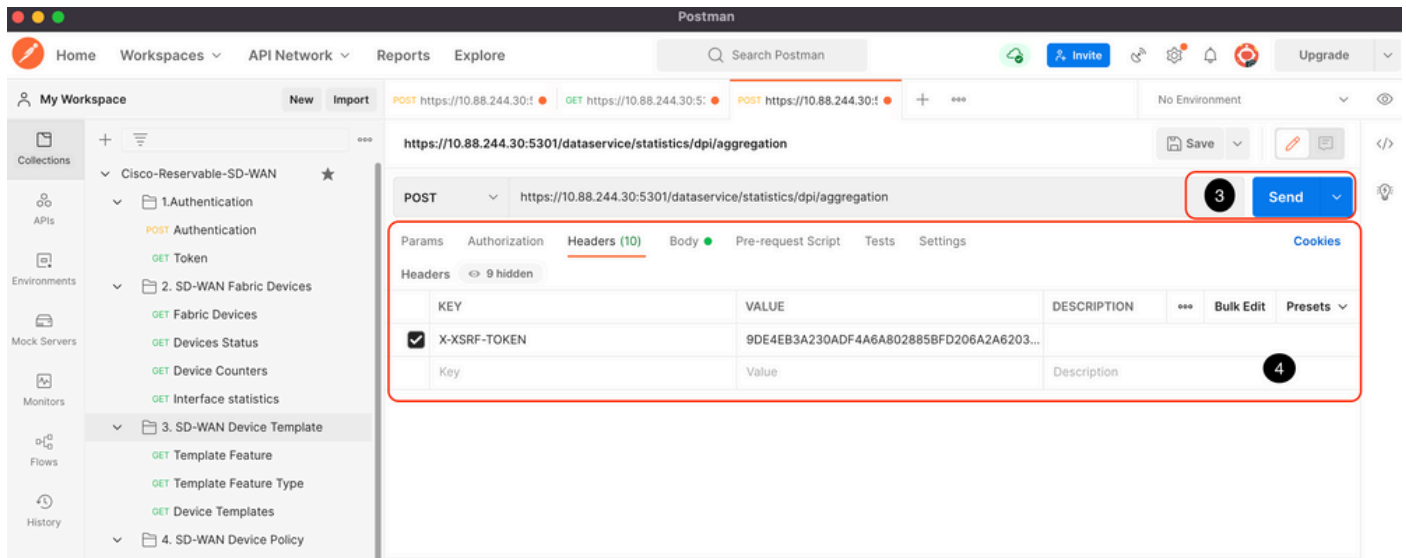
Nota: Esta llamada API contiene un cuerpo en formato JSON



3. Haga clic en **Encabezado** y agregue como **Clave** la cadena **X-XSRF-TOKEN** como valor.

4. Haga clic en **Enviar**.

La imagen mostrada muestra cómo debe aparecer la llamada a la API.



Llamada API de agregación DPI.

Paso 5. Cierre de la sesión

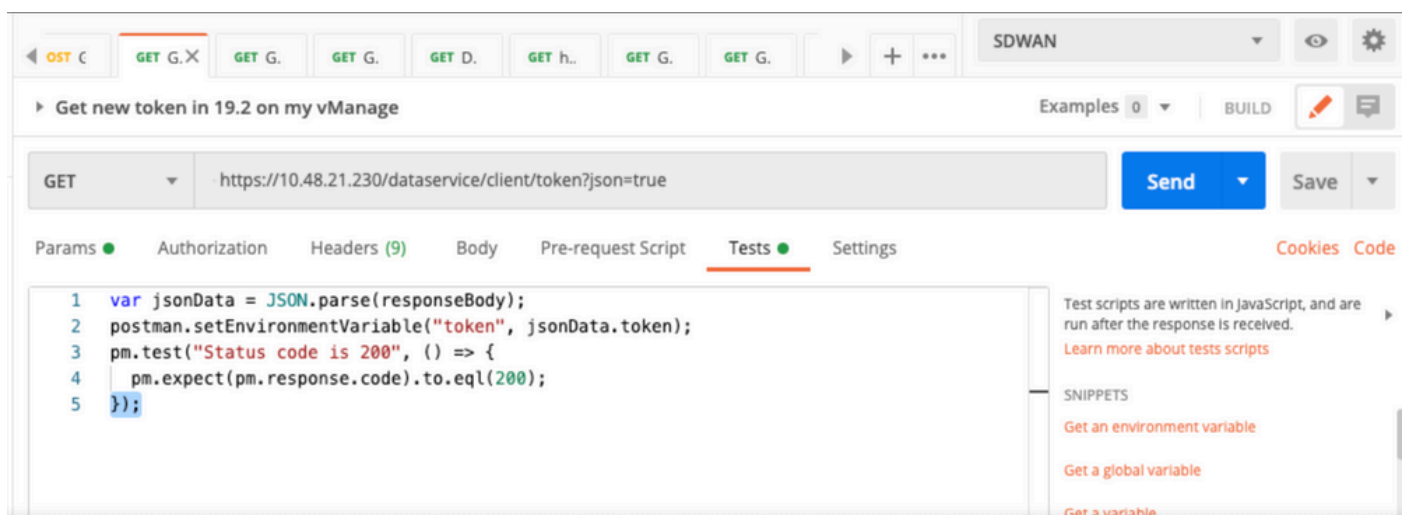
Una vez que haya recuperado toda la información necesaria de vManage o de los dispositivos, libere los recursos de vManage y elimine la posibilidad de que los usuarios malintencionados utilicen la sesión.

Ejecución de llamadas API en un entorno automatizado

Guardar cookies y variables para usarlas en llamadas API posteriores

¿Cómo guardar el token en una variable?

Guarde el token en una variable para volver a utilizarlo posteriormente.



Guarde el token en una variable

Cuando solicitemos el token en formato JSON, guárdelo. Utilice la pestaña **Pruebas** y pegue las líneas mostradas.

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

Posteriormente, cualquier llamada API puede utilizar una variable de token.

Key	Value	Description
<input checked="" type="checkbox"/> Host	<calculated when request is sent>	
<input checked="" type="checkbox"/> User-Agent	PostmanRuntime/7.26.3	
<input checked="" type="checkbox"/> Accept	*/*	
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate, br	
<input checked="" type="checkbox"/> Connection	keep-alive	
<input checked="" type="checkbox"/> X-XSRF-TOKEN	{{token}}	
<input checked="" type="checkbox"/> Content-Type	application/json	

Utilizar la variable de token

¿Cómo se borra la cookie SESSIONID para las nuevas sesiones?

Siempre que ejecute la llamada API para salir de, utilice JSESSIONID.

No podemos utilizar ninguna autenticación básica como hicimos en las versiones anteriores. En su lugar, solo proporcionamos credenciales y guardamos la ID en nuestra cookie. Antes de esto, podemos usar una prueba previa para borrar todas las cookies o cookies específicas.

```
1 const jar = pm.cookies.jar();  
2  
3 jar.clear(pm.request.url, function (error) {  
4   // error - <Error>  
5 });
```

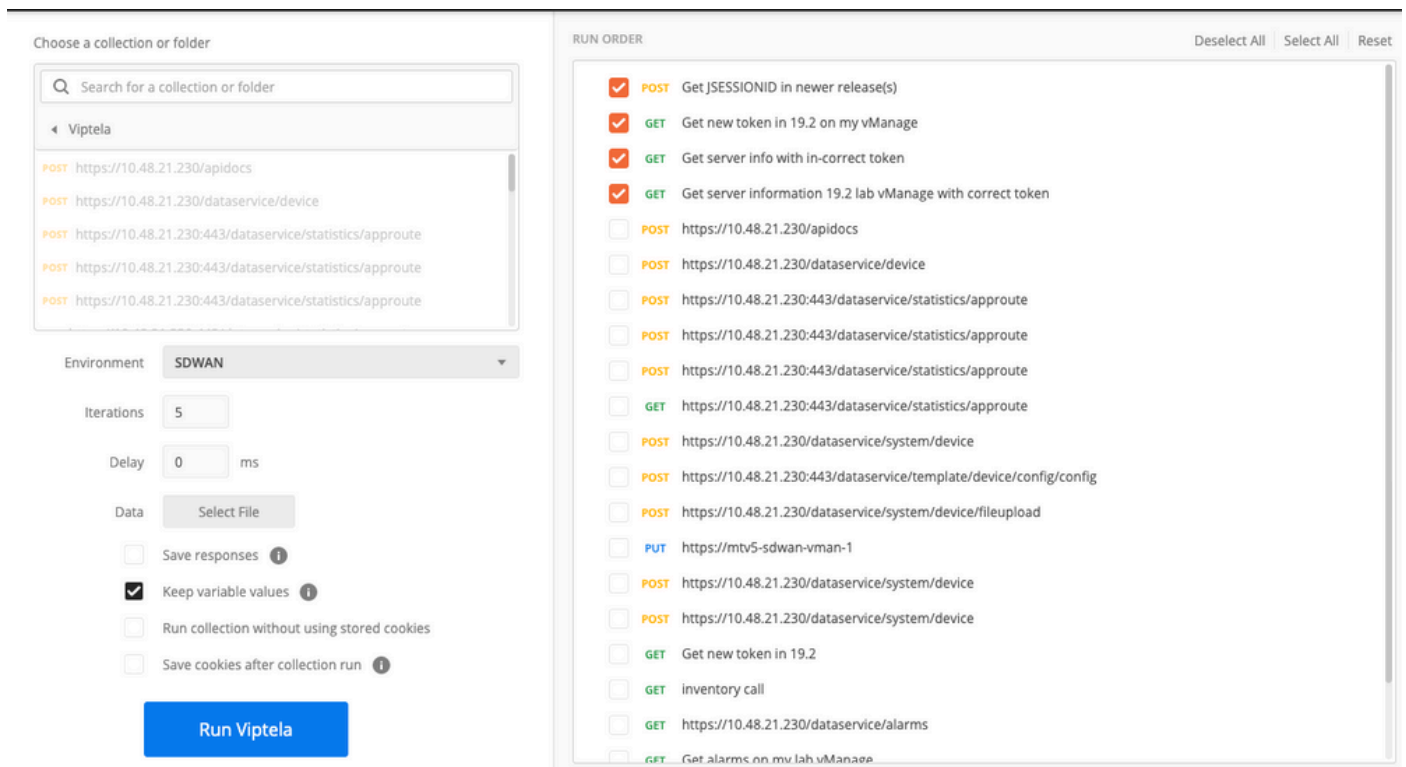
Borrar cookies

Esto es a través del código puesto en la secuencia de comandos Pre-request.

Cómo utilizar Collection Runner

Ahora que tenemos un entorno en el que podemos ejecutar sesiones y guardar datos específicos de cada sesión, puede ejecutar una secuencia de llamadas mediante Collection Runner.

Seleccione el orden de los eventos que desea repetir, seleccione el número de repeticiones para que Postman pueda ejecutar las llamadas API, el número de veces seleccionado con resultados por ejecución.



The screenshot displays the Postman Collection Runner interface. On the left, under 'Choose a collection or folder', a search bar contains 'Viptela'. Below it, a list of requests is shown, all with a 'POST' method and various URLs. The 'Environment' is set to 'SDWAN', 'Iterations' is 5, and 'Delay' is 0 ms. There are checkboxes for 'Save responses', 'Keep variable values' (checked), 'Run collection without using stored cookies', and 'Save cookies after collection run'. A blue 'Run Viptela' button is at the bottom. The right panel, 'RUN ORDER', shows a list of requests with checkboxes and colored labels (POST, GET, PUT) indicating their order and status. The first four requests are checked and have colored labels: POST (orange), GET (green), GET (green), and GET (green). The rest are unchecked.

Colección Runner

Desde la "biblioteca" de la llamada, colóquelos en un orden determinado para que se ejecute un flujo/orden específico.

Ponga en un resultado comprobar si obtiene un 200 OK u otro valor como respuesta y tratarlo como aprobado o suspenso.

The screenshot shows the Postman interface for a REST client. The request is a GET method to the URL `https://10.48.21.230/dataservice/client/token?json=true`. The 'Tests' tab is active, containing the following JavaScript code:

```
1 var jsonData = JSON.parse(responseBody);
2 postman.setEnvironmentVariable("token", jsonData.token);
3 pm.test("Status code is 200", () => {
4   pm.expect(pm.response.code).to.eql(200);
5 });
```

The response is shown in the 'Body' tab, with a status of 200 OK, a time of 67 ms, and a size of 550 B. The response body is displayed in JSON format:

```
1 {
2   "token": "23AE920117579F0EF9D470C2DE837A74C292D6A5929E098E06AB6358D399A61BD99B23D17D836D36EE0BAF764E1B10D52059"
3 }
```

Comprobar código de respuesta

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

Entonces podemos ver pasado o fallar en nuestras carreras.

Collection Runner Run Results My Workspace Run In Command Line Docs

20 PASSED 0 FAILED Viptela SDWAN just now

Run Summary Export Results Retry New

Iteration 1

- POST Get JSESSIONID in newer release(s) https://10.48.21.230/j_se... Viptela / Get JSESSIONID in newer ...
 - Status code is 200
- GET Get new token in 19.2 on my vManage https://10.48.21.230/dat... Viptela / Get new token in 19.2 on...
 - Status code is 200
 - 200 OK 53 ms 550 B
- GET Get server info with in-correct token https://10.48.21.230/dat... Viptela / Get server info with in-co...
 - Status code is 403
 - 403 Forbidden 56 ms 583 B
- GET Get server information 19.2 lab vManage with correct token https://10.48.21.230/dat... Viptela / Get server information 1...
 - Status code is 200
 - 200 OK 49 ms 486 B

Iteration 2

- POST Get JSESSIONID in newer release(s) https://10.48.21.230/j_se... Viptela / Get JSESSIONID in newer ...
 - Status code is 200
- GET Get new token in 19.2 on my vManage https://10.48.21.230/dat... Viptela / Get new token in 19.2 on...
 - Status code is 200
 - 200 OK 48 ms 550 B
- GET Get server info with in-correct token https://10.48.21.230/dat... Viptela / Get server info with in-co...
 - Status code is 403
 - 403 Forbidden 49 ms 583 B

Console

Ejecución automatizada

Acerca de esta traducción

Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).