Apigee Lab 2a: Route Rules and the Debug Tool

experiment Lab schedule 1 hour 30 minutes universal_currency_alt No cost

show_chart Introductory

Overview

In this lab, you will create two target endpoints and conditionally route to them using route rules. You also learn how to troubleshoot your proxies using the Apigee Debug tool.

Objectives

In this lab, you will learn how to perform the following tasks:

- Create multiple target endpoints within a single proxy.
- Modify route rules to conditionally route to the target endpoints.
- Trace API calls flowing through a proxy.
- Understand the information available in a debug session.
- Set filters to capture only matching calls.

Setup and requirements

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.

2. Note the lab's access time (for example, `1:15:00`), and make sure you can finish within that time. There is no pause feature. You can restart if needed, but you have to start at the beginning.

3. When ready, click **Start lab**.

4. Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.

5. Click **Open Google Console**.

6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts. If you use other credentials, you'll receive errors or **incur charges**.

7. Accept the terms and skip the recovery resource page.

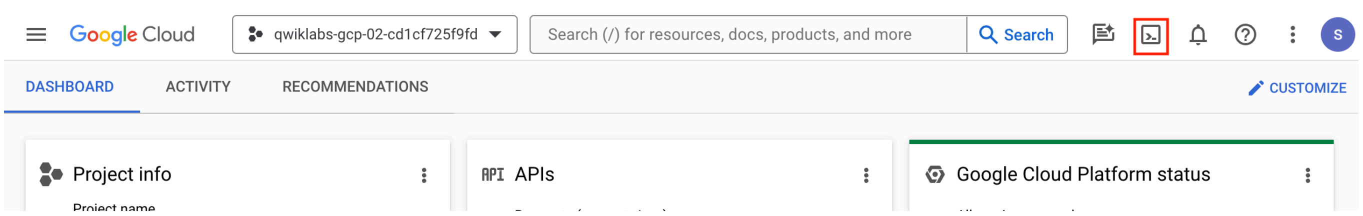
Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud.

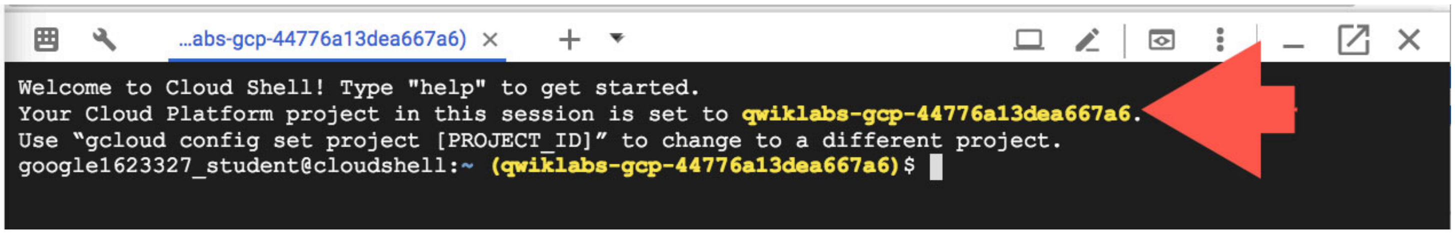
Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.


Google Cloud
qwiklabs-gcp-02-cd1cf725f9fd
Search (/) for resources, docs, products, and more
Search
DASHBOARD
ACTIVITY
RECOMMENDATIONS
CUSTOMIZE
Project info
APIs
Google Cloud Platform status

2. Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6)$
```

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

- You can list the active account name with this command:

```
gcloud auth list
```

content_c

Output:

```
Credentialed accounts:
 - @.com (active)
```

Example output:

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

- You can list the project ID with this command:

```
gcloud config list project
```

content_c

Output:

```
[core]
project =
```

Example output:

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

Note: Full documentation of **gcloud** is available in the gcloud CLI overview guide .

Task 1. Create a new API proxy

In this task, you create a new API proxy that calls the retail backend.

1. In the Google Cloud console, on the **Navigation menu** (☰), select **Integration Services > Apigee > Proxy Development > API proxies**.

2. To start the proxy wizard, click **+Create**.

3. Leave **Proxy template** unchanged.

4. Specify the following settings:

Property	**Value**
Proxy Name	**lab2a-v1**
Base path	**/lab2a/v1**
Target (Existing API)	**https://gcp-cs-training-01-test.apigee.net/training/db**

Note: Confirm that you are using `/lab2a/v1` for the base path, and not `/lab2a-v1`.

5. Click **Create**.

6. Click the **Develop** tab.

Task 2. Add a new target endpoint and route rule

In this task, you create a new target endpoint and route rule to call a new backend service.

1. In the Navigator pane, click the **Add target endpoint** (+) button for **Target endpoints**.

2. In the **Add target endpoint** pane, specify the following values:

Property	Value
Target endpoint name	**uuid_service**
HTTP target	**https://httpbin.org/uuid**

   This service returns a UUID.

   Note: UUIDs, or Universally Unique Identifiers, are used to create globally unique identifiers without requiring a centralized authority to provide them. These are also known as GUIDs, or Globally Unique Identifiers.

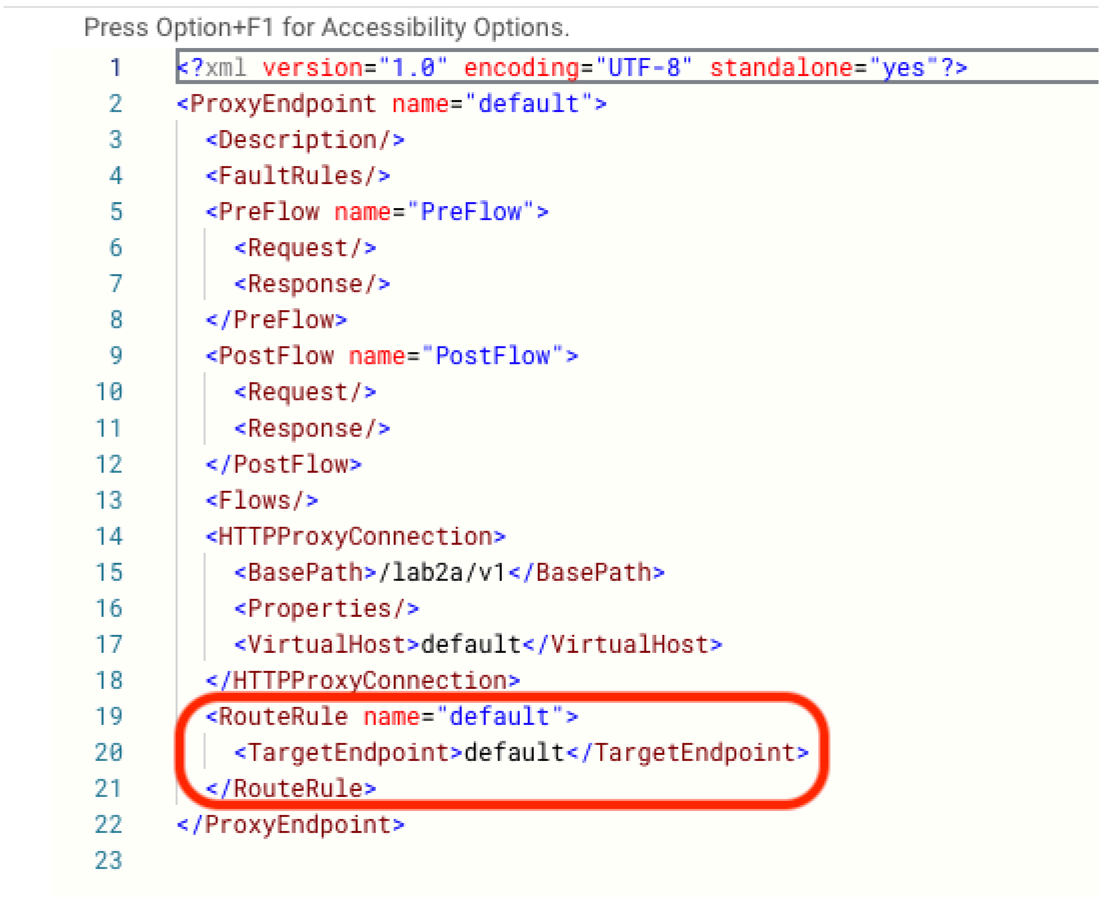
3. Click **Add**.

   The `uuid_service` target endpoint appears in the code pane. Functionality that is specific to the target should be placed in the target endpoint flows.

4. In the Navigator pane, click **Proxy endpoints** > **default**.

   The default ProxyEndpoint configuration is displayed in the code box. Examine the `RouteRule` section in the proxy endpoint.

default.xml



Press Option+F1 for Accessibility Options.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProxyEndpoint name="default">
  <Description/>
  <FaultRules/>
  <PreFlow name="PreFlow">
    <Request/>
    <Response/>
  </PreFlow>
  <PostFlow name="PostFlow">
    <Request/>
    <Response/>
  </PostFlow>
  <Flows/>
  <HTTPProxyConnection>
    <BasePath>/lab2a/v1</BasePath>
    <Properties/>
    <VirtualHost>default</VirtualHost>
  </HTTPProxyConnection>
  <RouteRule name="default">
    <TargetEndpoint>default</TargetEndpoint>
  </RouteRule>
</ProxyEndpoint>
```

When we created the proxy, it had only a single target endpoint. The single route rule routes all traffic to the **default** target endpoint because there is no condition for the route rule. We need to add a route rule to conditionally route to the `uuid_service` target endpoint.

5. Replace the following:

```
<RouteRule name="default">
  <TargetEndpoint>default</TargetEndpoint>
</RouteRule>
```

with:

```
<RouteRule name="uuid">
  <Condition>proxy.pathsuffix == "/uuid"</Condition>
  <TargetEndpoint>uuid_service</TargetEndpoint>
</RouteRule>
<RouteRule name="default">
```

content_c

```
    <TargetEndpoint>default</TargetEndpoint>
  </RouteRule>
```

This new route rule routes to the `uuid_service` target endpoint if the proxy's path suffix is **/uuid**.

6. Click **Save**. If you are notified that the proxy was saved as a new revision, click **OK**.

7. Click **Deploy**.

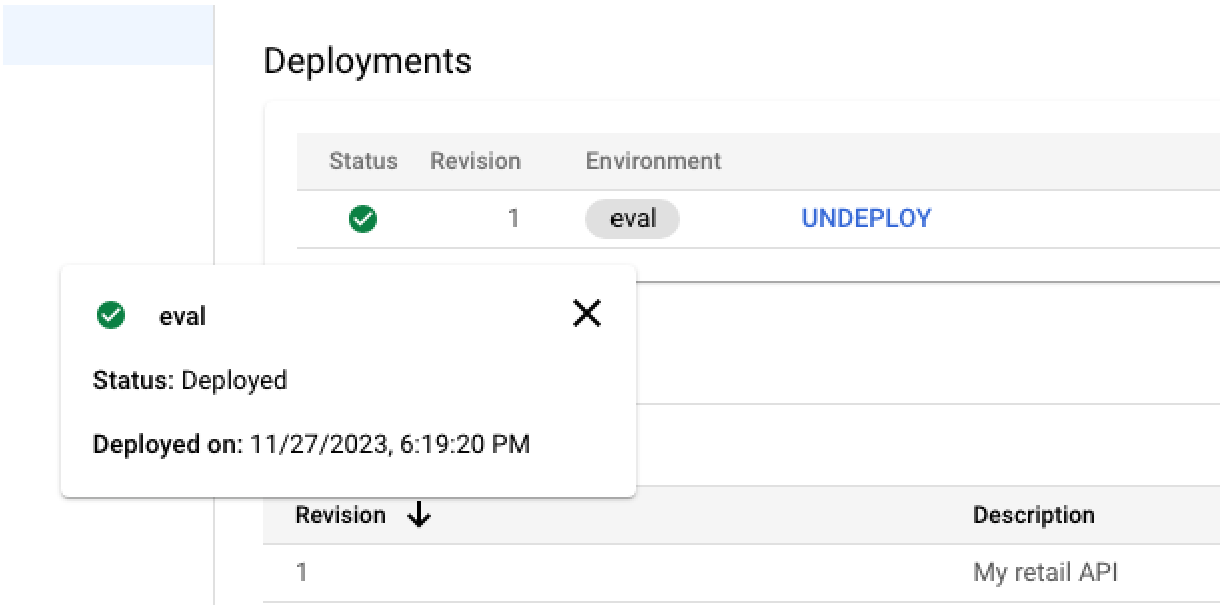
8. To specify that you want the new revision deployed to the eval environment, select **eval** as the **Environment**, and then click **Deploy**.

   Note: Leave the "Service Account" field empty.

9. Click **Confirm**.

Check deployment status

A proxy that is deployed and ready to take traffic will show a green status.



Status	Revision	Environment
✓	1	eval

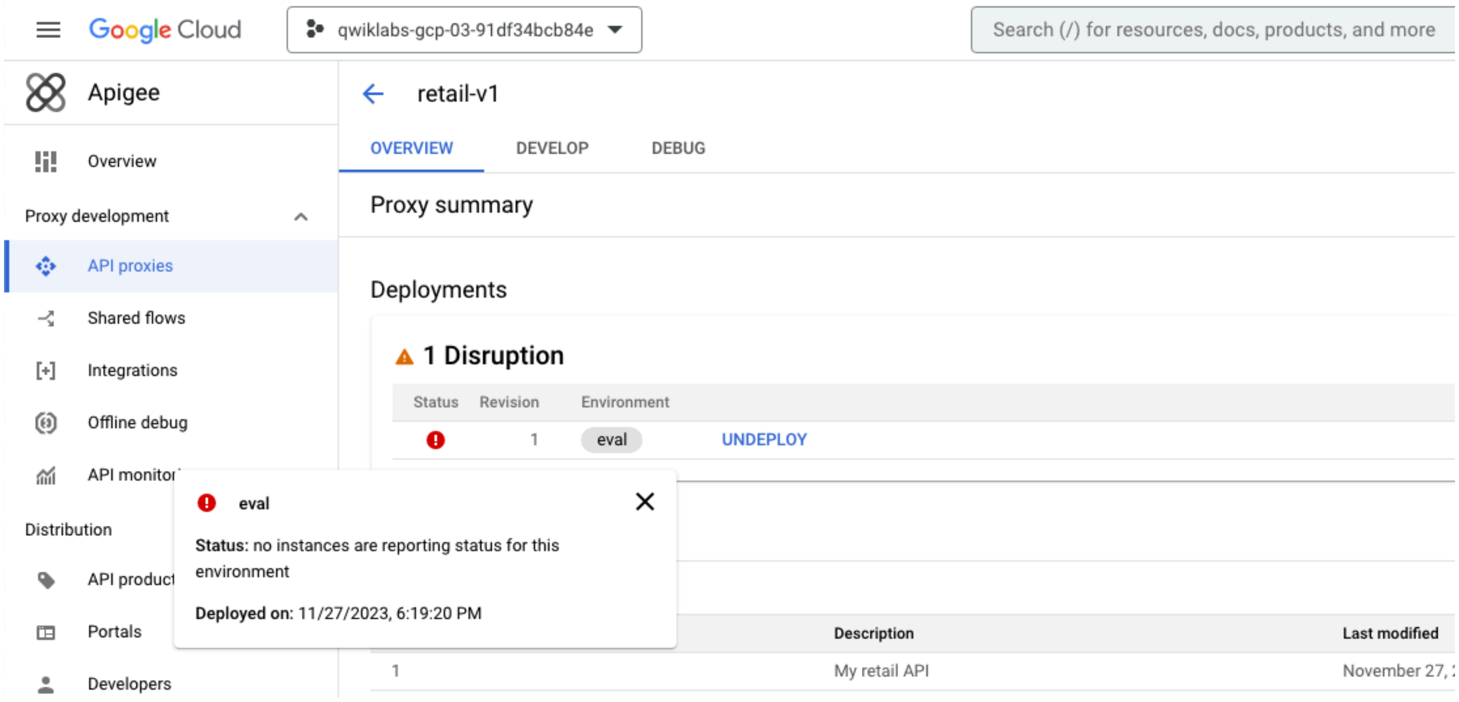
✓ eval

Status: Deployed

Deployed on: 11/27/2023, 6:19:20 PM

Revision	Description
1	My retail API

When a proxy is marked as deployed but the runtime is not yet available and the environment is not yet attached, you may see a red warning sign. Hold the pointer over the **Status** icon to see the current status.



Deployments

⚠ 1 Disruption

Status	Revision	Environment	
●	1	eval	UNDEPLOY

eval

Status: no instances are reporting status for this environment

Deployed on: 11/27/2023, 6:19:20 PM

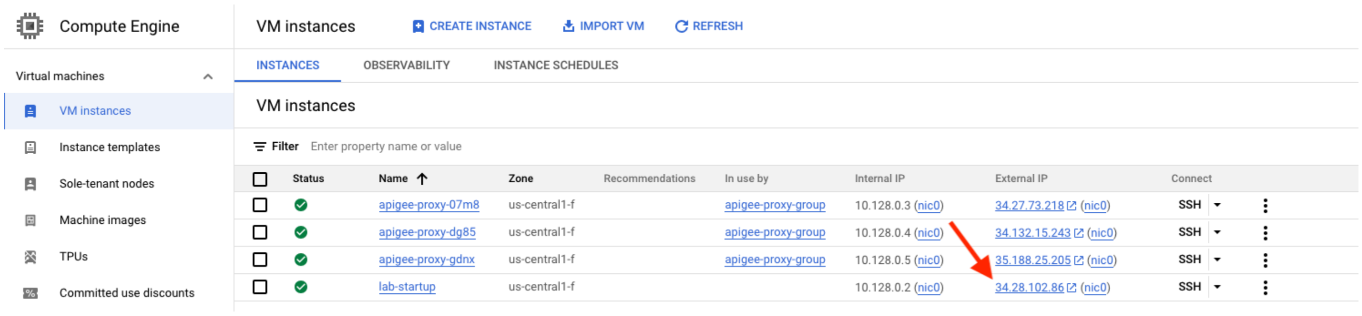
	Description	Last modified
1	My retail API	November 27,

If the proxy is deployed and shows as green, your proxy is ready for API traffic. If your proxy is not deployed because there are no runtime pods, you can check the status of provisioning.

Check provisioning dashboard

1. In the Google Cloud Console, navigate to **Compute Engine > VM instances**.

2. To open the Lab Startup Tasks dashboard, click on the **External IP** for the **lab-startup** VM.



Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	apigee-proxy-07m8	us-central1-f		apigee-proxy-group	10.128.0.3 (nic0)	34.27.73.218 (nic0)	SSH
✓	apigee-proxy-dg85	us-central1-f		apigee-proxy-group	10.128.0.4 (nic0)	34.132.15.243 (nic0)	SSH
✓	apigee-proxy-gdnx	us-central1-f		apigee-proxy-group	10.128.0.5 (nic0)	35.188.25.205 (nic0)	SSH
✓	lab-startup	us-central1-f			10.128.0.2 (nic0)	34.28.102.86 (nic0)	SSH

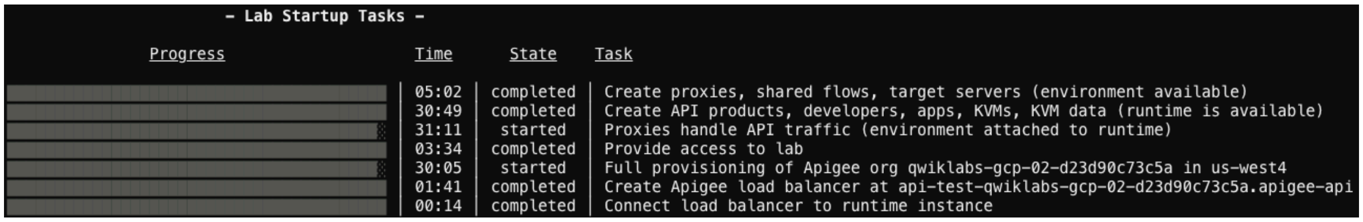
3. If you see a redirect notice page, click the link to the external IP address.

   A new browser window will open. Lab startup tasks are shown with their progress.

   - *Create proxies, shared flows, target servers* should be complete when you first enter the lab, allowing you to use the Apigee console for tasks like proxy editing.

- *Create API products, developers, apps, KVMs, KVM data* indicates when the runtime is available and those assets may be saved.

- *Proxies handle API traffic* indicates when the eval environment has been attached to the runtime and the deployed proxies can take runtime traffic.



```
                - Lab Startup Tasks -

          Progress                Time      State    Task

                                | 05:02 | completed | Create proxies, shared flows, target servers (environment available)
                                | 30:49 | completed | Create API products, developers, apps, KVMs, KVM data (runtime is available)
                                | 31:11 |  started  | Proxies handle API traffic (environment attached to runtime)
                                | 03:34 | completed | Provide access to lab
                                | 30:05 |  started  | Full provisioning of Apigee org qwiklabs-gcp-02-d23d90c73c5a in us-west4
                                | 01:41 | completed | Create Apigee load balancer at api-test-qwiklabs-gcp-02-d23d90c73c5a.apigee-api
                                | 00:14 | completed | Connect load balancer to runtime instance
```

In this case, you need to wait for *Proxies handle API traffic* to complete.

While you are waiting

Take note of the following:

- Refer to the Endpoint properties reference documentation on endpoint properties that can be set for target and proxy endpoints.
- Refer to the Understanding routes documentation on how route rules are used to select a target endpoint.

Task 3. Test the API proxy

In this task, you use curl to call each service.

1. In **Cloud Shell**, send the following curl command:

```
curl -i -X GET "https://api-test-${GOOGLE_CLOUD_PROJECT}.apiservice
```

content_c

The curl command responds with a JSON array of category objects.

2. Now, send the following curl command:

```
curl -i -X GET "https://api-test-${GOOGLE_CLOUD_PROJECT}.apiservice
```

content_c

When you changed the proxy suffix to /uuid, you probably expected the uuid_service target to be called. Instead, you will see an HTML error.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server.  If you entered the
URL manually please check your spelling and try again.</p>
```

3. To directly call the backend service you were intending to call, open another tab in your browser, and navigate to the URL you used in the `uuid_service` target endpoint:

```
https://httpbin.org/uuid
```

content_c

Opening a web page uses a **GET** verb, and this URL successfully returns a JSON payload with a UUID.

Something must be going wrong. How do you determine what is happening?

Task 4. Use the debug tool to debug the issue

In this task, you explore how to use the debug tool to debug API proxy issues.

1. Select the **Debug** tab.

2. Click **Start Debug Session**.

3. In the **Start debug session** pane, on the Environment dropdown, select **eval**.

   The deployed revision number will also show in the dropdown.

4. Click **Start**.

   A debug session will last a maximum of 10 minutes or until 15 transactions are received, whichever comes first.

5. Back in Cloud Shell, send the following curl command:

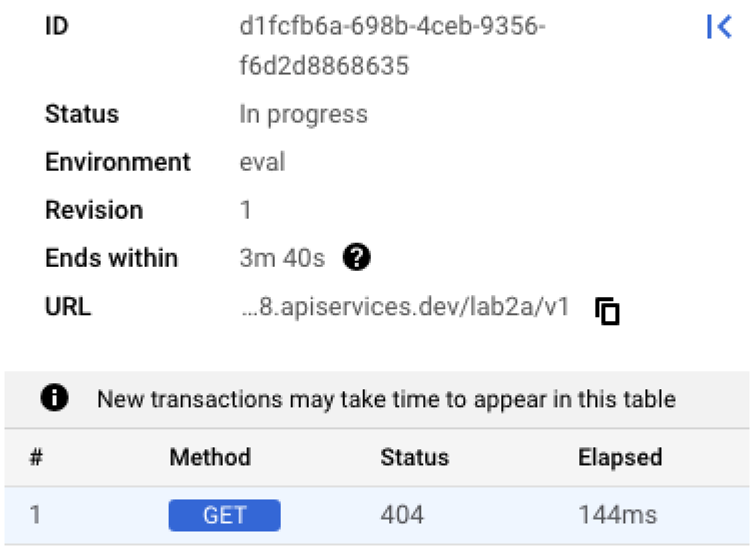
   ```
   curl -i -X GET "https://api-test-${GOOGLE_CLOUD_PROJECT}.apiservice
   ```

   content_c

   A transaction for this request should appear in the Transactions pane on the left. When a transaction is selected, you'll see a trace of the request and response through Apigee.

   You should see a **404** Status if your backend URL was correctly set and you correctly sent the curl command.

   - The debug session details pane shows a list of transactions that have been received since the debug session was started.



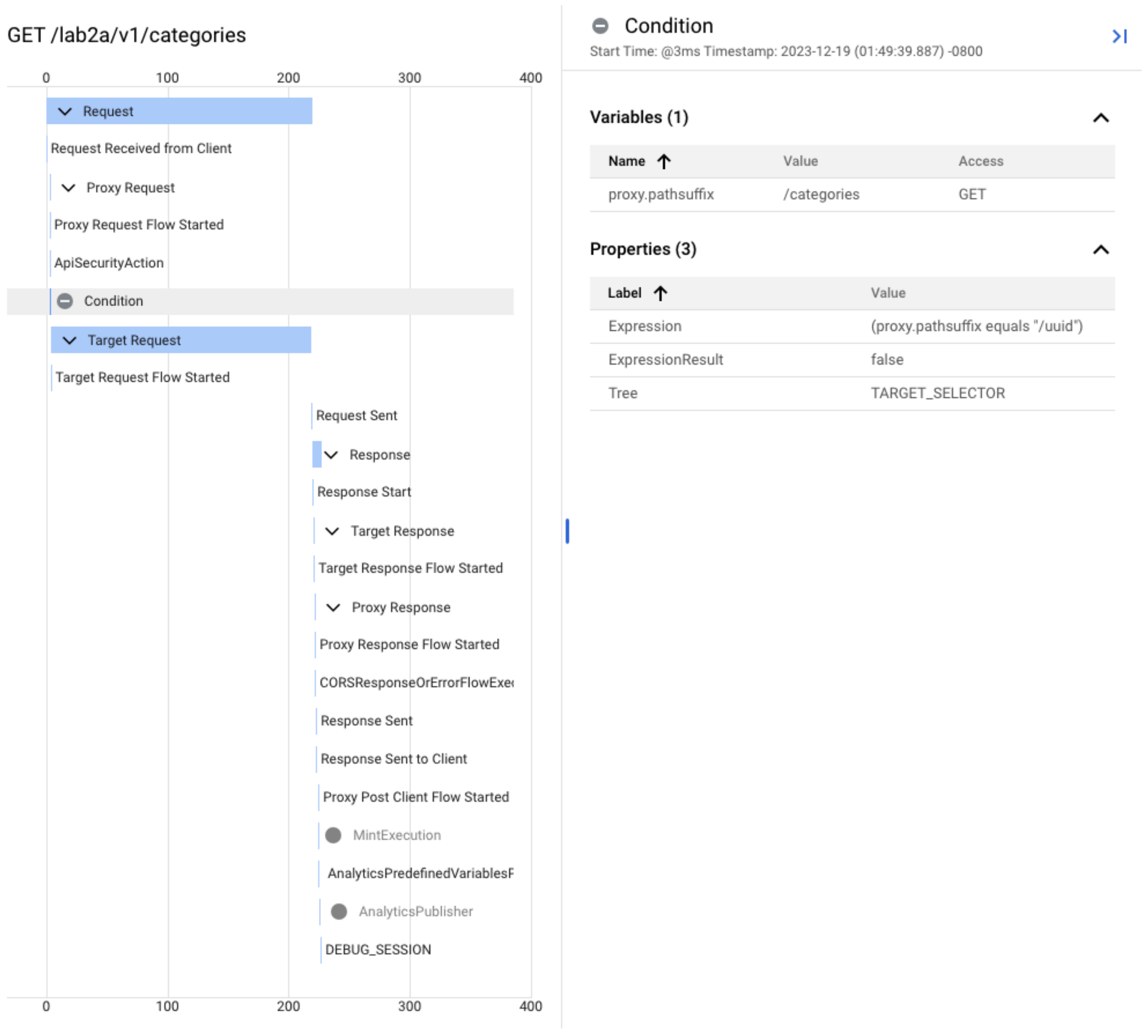
ID	d1fcfb6a-698b-4ceb-9356-f6d2d8868635
Status	In progress
Environment	eval
Revision	1
Ends within	3m 40s
URL	...8.apiservices.dev/lab2a/v1

New transactions may take time to appear in this table

#	Method	Status	Elapsed
1	GET	404	144ms

- The transaction Gantt chart pane shows a Gantt chart view of all policies and flows evaluated during the execution of the selected transaction. The horizontal axis of the diagram denotes the time at which each step occurred, measured in milliseconds. Each step is represented by a rectangle that extends from the start time to the end time of the step. You can step through a debug session by using the **Back** and **Next** buttons.

When you select a step on the Gantt chart, the details of that step show up in the details pane. For example, if you click on a policy, you would see details like variables read and updated by the policy, as well as properties set in the policy configuration.



Condition

Start Time: @3ms Timestamp: 2023-12-19 (01:49:39.887) -0800

Variables (1)

Name	Value	Access
proxy.pathsuffix	/categories	GET

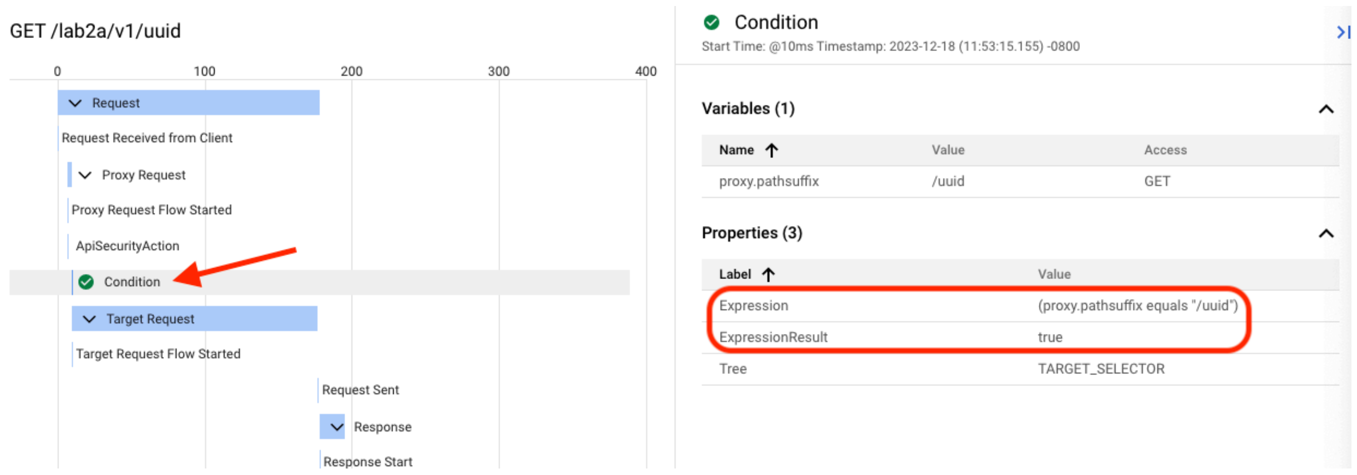
Properties (3)

Label	Value
Expression	(proxy.pathsuffix equals "/uuid")
ExpressionResult	false
Tree	TARGET_SELECTOR

Exploring the transaction

1. Click the `GET 404` transaction.

2. In the Gantt map, in the request, click on **Condition**.



Condition

Start Time: @10ms Timestamp: 2023-12-18 (11:53:15.155) -0800

Variables (1)

Name	Value	Access
proxy.pathsuffix	/uuid	GET

Properties (3)

Label	Value
Expression	(proxy.pathsuffix equals "/uuid")
ExpressionResult	true
Tree	TARGET_SELECTOR

This shows the details of the condition you added in the first route rule.

The condition **Expression** is `proxy.pathsuffix equals "/uuid"`, and the **ExpressionResult** is `true`.

The true result indicates that your request was handled by the `uuid_service` target endpoint.

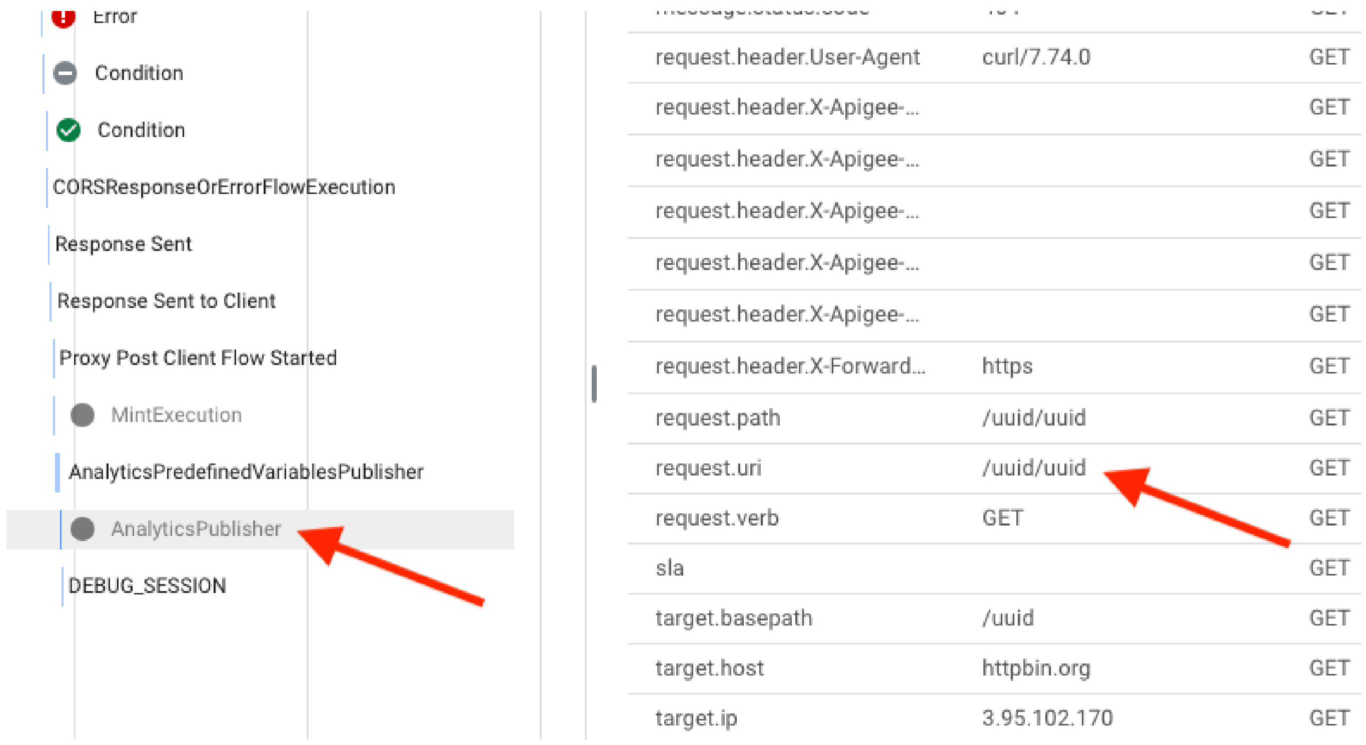
3. Click **Response Start**.

   The response content shows that the backend server returned 404 Not Found. You need to determine what URL was called on the backend server.

4. Click **AnalyticsPublisher**.

   Apigee captures analytics for the API calls that travel through your proxies. The **AnalyticsPublisher** step in the transaction details shows the variables that were captured.

   The variable **request.uri** contains the path that was called by the Apigee proxy, `/uuid/uuid`.



Error

Condition

Condition

CORSResponseOrErrorFlowExecution

Response Sent

Response Sent to Client

Proxy Post Client Flow Started

MintExecution

AnalyticsPredefinedVariablesPublisher

AnalyticsPublisher

DEBUG_SESSION

request.header.User-Agent	curl/7.74.0	GET
request.header.X-Apigee-...		GET
request.header.X-Apigee-...		GET
request.header.X-Apigee-...		GET
request.header.X-Apigee-...		GET
request.header.X-Apigee-...		GET
request.header.X-Forward...	https	GET
request.path	/uuid/uuid	GET
request.uri	/uuid/uuid	GET
request.verb	GET	GET
sla		GET
target.basepath	/uuid	GET
target.host	httpbin.org	GET
target.ip	3.95.102.170	GET

You can probably see the issue now. The request to the backend service was a **GET** on **https://httpbin.org/uuid/uuid**, and the backend service returned the HTML error. The backend service URL only requires a single uuid. Why did this happen?

By default, the path suffix is automatically added to the target endpoint URL. This is why a request to `/lab2a/v1/categories` went to the categories endpoint in the retail backend service. `/categories` was the path suffix, which was automatically added to the target endpoint URL.

In this request, `/uuid` follows the base path, and is therefore the path suffix. It was being added to the target endpoint URL, which already ends with `/uuid`.

You could remove `/uuid` from the target endpoint URL. Instead, you will change the proxy to not automatically append the path suffix to the target endpoint URL.

Task 5. Update the API proxy

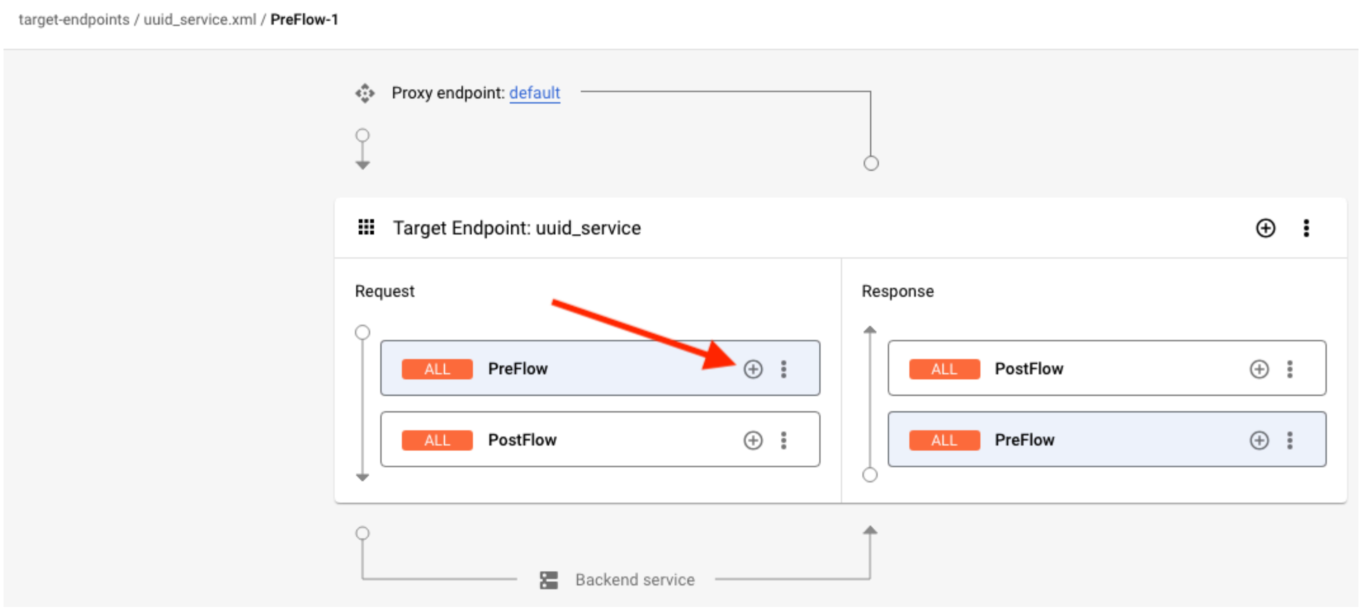
In this task, you add a policy to stop the path suffix from being added to the target service URL.

1. Click the **Develop** tab.

2. In the Navigator pane, click **Target endpoints > uuid_service > PreFlow**.

   A visual representation of the target endpoint preflow is displayed.

3. On the **Request PreFlow**, click **Add Policy Step (+)**.



   This policy must run before we call the backend, so the step must be in the request path.

4. In the **Add policy step** pane, select **Create new policy**, and then select **Extension > Javascript**.

5. Specify the following values:

Property	Value
Name	**JS-DisablePathSuffixCopy**
Display name	**JS-DisablePathSuffixCopy**

6. For **Javascript file**, select **Create New Resource**.

7. For **Resource name**, specify `DisablePathSuffixCopy.js`.

8. Click **Add**.

9. For **Javascript file**, select **DisablePathSuffixCopy.js**.

10. Click **Add**.

11. In another browser tab, open the Flow variables reference page.

    This page lists the available built-in variables and is very useful when building API proxies. In the **target** section, there is a variable named **target.copy.pathsuffix**. Setting the variable to false will block the pathsuffix from being automatically copied to the target request URL. You can also specify the backend URL directly using the **target.url** variable.

12. In the Navigator pane, click **Policies > JS-DisablePathSuffixCopy**.

    The policy XML configuration is shown in the Code pane.

    You typically add JavaScript code to the *.js* file, but in this case the code is a single line, so we can add it directly in the JavaScript configuration.

13. Replace the current JavaScript configuration:

```
<Javascript continueOnError="false" enabled="true" timeLimit="200" name="JS
  <DisplayName>JS-DisablePathSuffixCopy</DisplayName>
  <Properties/>
  <ResourceURL>jsc://DisablePathSuffixCopy.js</ResourceURL>
</Javascript>
```

    with:

```
<Javascript continueOnError="false" enabled="true" timeLimit="200"
  <Source>
    context.setVariable("target.copy.pathsuffix", false);
  </Source>
</Javascript>
```

content_c

    The single line of JavaScript sets the **target.copy.pathsuffix** variable to false.

14. The *.js* file is no longer needed, so click **Resources > jsc > DisablePathSuffixCopy.js**, and then click **Delete Resource** (🗑).

15. Confirm the deletion of the script by clicking **Delete**.

16. To save the updates, click **Save**, and then click **Save as New Revision**.

17. Click **Deploy**.

18. To specify that you want the new revision deployed to the eval environment, click **Deploy**, and then click **Confirm**.

Task 6. Test the updated proxy

In this task, you verify that the proxy correctly calls the UUID service.

1. Click the **Debug** tab, and then click **Start Debug Session**.

2. In the **Start debug session** pane, on the Environment dropdown, select **eval**.

3. For **Filter**, select **Custom**.

   For this debug session, you will specify a custom filter. Requests that do not match the filter will not be captured. This feature is very important when trying to debug the proxy in production, because the vast majority of the API traffic may not be what you are trying to capture.

4. For Value, specify this filter:

   ```
   proxy.pathsuffix == "/uuid"
   ```

   content_c

   This filter will only capture calls that are going to the UUID service.

5. Click **Start**.

6. In Cloud Shell, send the following curl command a few times:

```
curl -i -X GET "https://api-test-${GOOGLE_CLOUD_PROJECT}.apiservice
```

content_c

   You should see a UUID in the curl response.

7. Send the following curl command a few times too:

```
curl -i -X GET "https://api-test-${GOOGLE_CLOUD_PROJECT}.apiservice
```

content_c

   You should see the categories in the curl response.

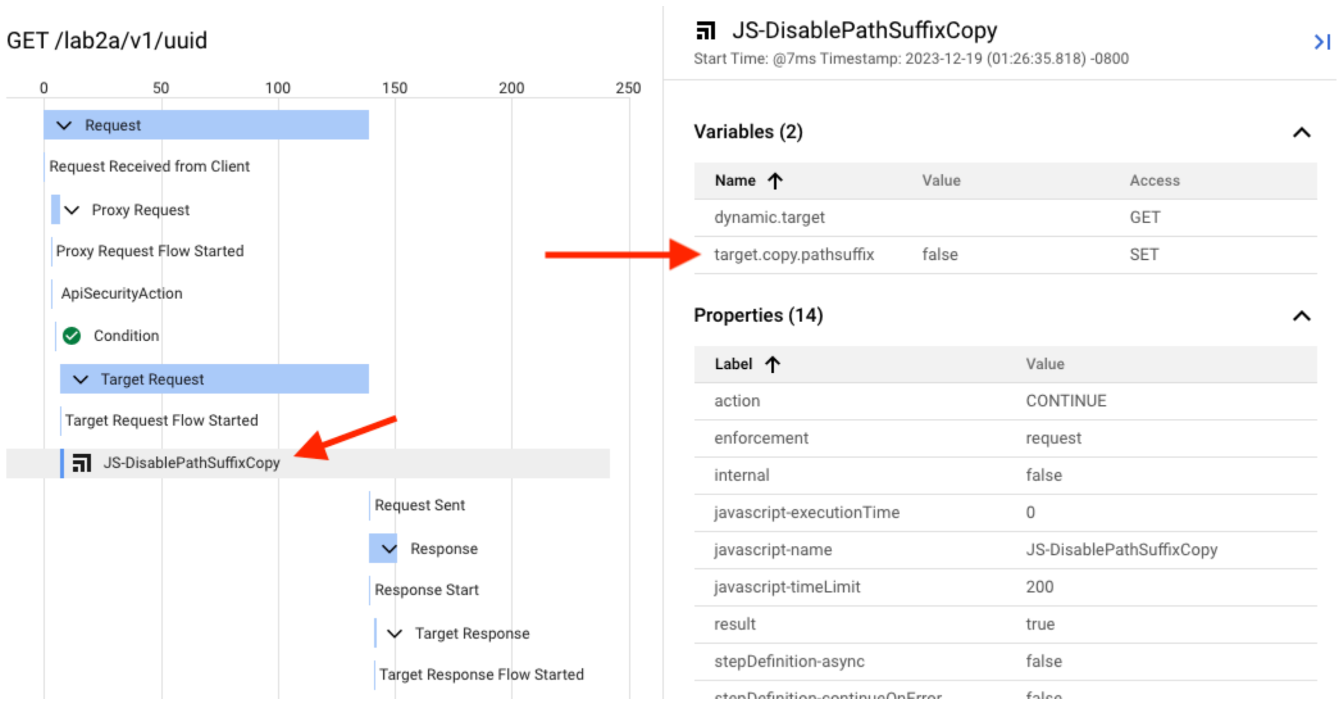
8. Return to the debug tool.

   You should only see UUID calls, because the categories calls were filtered out.

9. Click one of the transactions.

10. Click on the **JS-DisablePathSuffixCopy** step.

    You can see in the policy details that **target.copy.pathsuffix** was set to false.



Name	Value	Access
dynamic.target		GET
target.copy.pathsuffix	false	SET

Label	Value
action	CONTINUE
enforcement	request
internal	false
javascript-executionTime	0
javascript-name	JS-DisablePathSuffixCopy
javascript-timeLimit	200
result	true
stepDefinition-async	false

Congratulations!

In this lab, you learned about route rules and using multiple targets. You also learned all about the debug tool. Now you should be able to debug issues with your proxies!

End your lab