

Overview

In this lab, you learn about handling faults in an Apigee proxy.

Objectives

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

- Rewrite an error for a fault generated by a policy.
- Manually raise a fault, specifying the error response.
- Allow only approved operations through to the target.

Setup

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**.

- Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
- Click **Open Google Console**.
- 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**.
- Accept the terms and skip the recovery resource page.

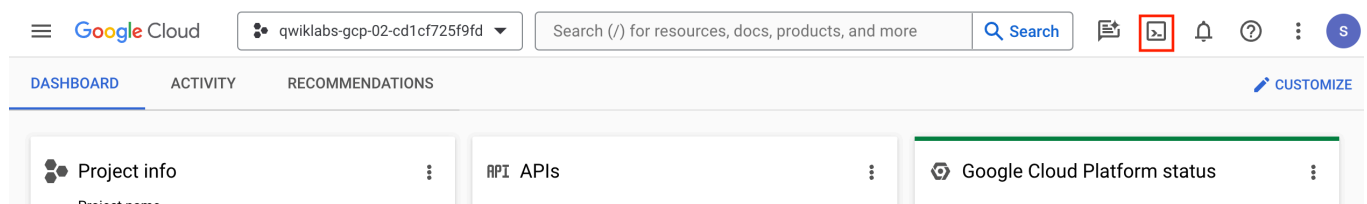
Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

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.

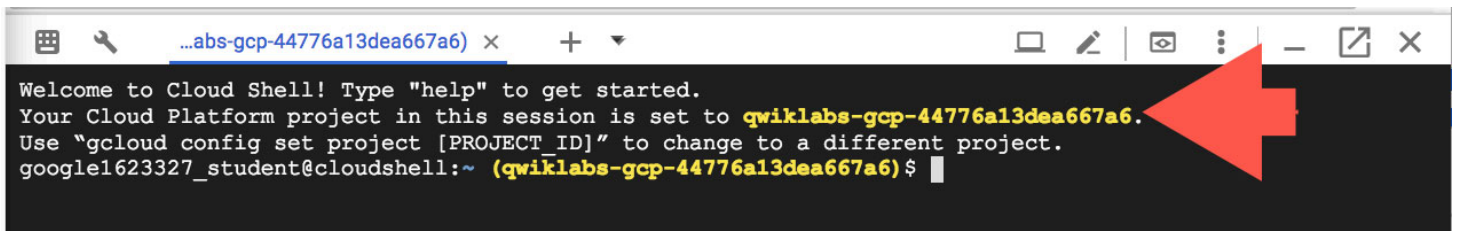
Google Cloud Shell provides command-line access to your Google Cloud resources.

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



- 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:



```
...abs-gcp-44776a13dea667a6) x + ▾
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 .

Preloaded assets

These assets have already been added to the Apigee organization:

- The **retail-v1** API proxy
- The oauth-v1 API proxy (for generating OAuth tokens)
- The backend-credentials shared flow (used by retail-v1)
- The TS-Retail target server in the eval environment (used by retail-v1)

These assets will be added to the Apigee organization as soon as the runtime is available:

- The API products, developer, and **developer app** (used by retail-v1)
- The ProductsKVM key value map in the eval environment (used by backend-credentials)
- The ProductsKVM key value map entries backendId and backendSecret

The **highlighted** items are used during this lab.

Note: Revision 1 of the retail-v1 proxy is marked as deployed, and is immutable. If you ever make a mistake in your proxy code that you can't recover from, you can select revision 1 and restart editing from there.

Task 1. Rewrite the error generated by the JSONThreatProtection policy

In this task, you add a FaultRule to handle a JSONThreatProtection policy fault and rewrite the error message.

Create a FaultRule

1. In the Google Cloud console, on the **Navigation menu** (≡), select **Integration Services > Apigee > Proxy Development > API proxies**.
2. Select the **retail-v1** proxy.
3. Click the **Develop** tab.

You are modifying the version of the retail-v1 proxy that was created during Labs 1 through 9.

4. In the Navigator pane, click **Proxy endpoints > default**.
5. Click on the **default** proxy endpoint in the Navigator.

In the **default.xml** pane, you see the XML representation of the default proxy endpoint. In the file you might see an empty *FaultRules* element: `<FaultRules/>`.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ProxyEndpoint name="default">
3   <Description/>
4   <FaultRules/>
5   <PreFlow name="PreFlow">
6     <Request>
7       <Step>
8         <Name>VA-VerifyKey</Name>
9       </Step>
10      <Step>
11        <Name>AM-RemoveAuth</Name>
12      </Step>
13    </Request>
14    <Response/>
15  </PreFlow>
16  <PostFlow name="PostFlow">
```



6. If you have an empty FaultRules element (<FaultRules/>) in default.xml, delete that line.

7. Create a new **FaultRules** element at the same location, inside the ProxyEndpoint element:

```
<FaultRules>
  <FaultRule name="JSONThreat">
    <Condition>jsonattack.JSONTP-Protect.failed ==
true</Condition>
  </FaultRule>
</FaultRules>
```

content_co

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <ProxyEndpoint name="default">
3    <Description/>
4    <FaultRules>
5      <FaultRule name="JSONThreat">
6        <Condition>jsonattack.JSONTP-Protect.failed == true</Condition>
7      </FaultRule>
8    </FaultRules>
9    <PreFlow name="PreFlow">
10     <Request>
```

This is a fault rule that is executed when the JSONThreatProtection policy raises a fault. The policy name in the condition (JSONTP-Protect) must exactly match the name of the JSONThreatProtection policy in your proxy. Fault variables that are generated by the JSONThreatProtection policy are listed in the documentation for the policy.

Rewrite the error message using an AssignMessage policy

The FaultRules flows do not show up in the graphical representation of the proxy endpoint, so you will need to create a policy without attaching it to a flow, and then add it to the XML.

1. In the Navigator pane, for **Policies**, click **Add Policy (+)**.

2. In the **Create policy** pane, for **Select policy**, select **Mediation > Assign Message**.

3. Specify the following values:

Property	Value
----------	-------

Name	AM-400JSONThreat
Display name	AM-400JSONThreat

4. Click **Create**.

5. Replace the AssignMessage configuration with:

```
<AssignMessage continueOnError="false" enabled="true" name="AM-400JSONThreat">
  <AssignVariable>
    <Name>jsonThreatErrorRegex</Name>
    <Value>JSONThreatProtection\[JSONTP-Protect\]: Execution failed. reason\: JSONThreatProtection\[JSONTP-Protect\]:
  </Value>
</AssignVariable>
<Set>
  <Payload contentType="application/json">{
    "error": "Invalid JSON payload:
    {replaceFirst(error.message,jsonThreatErrorRegex, '')}"
  }</Payload>
  <StatusCode>400</StatusCode>
  <ReasonPhrase>Bad Request</ReasonPhrase>
</Set>
<IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http"
type="response"/>
</AssignMessage>
```

content_c

This policy rewrites the error response that was generated by the JSONThreatProtection policy. The status code is set to **400 Bad Request**, which is more appropriate than the default status code provided by the policy (500 Internal Server Error).

The replaceFirst template inside the payload is used to strip off the beginning of the error message. **error.message** for this policy looks like this:

```
JSONThreatProtection[JSONTP-Protect]: Execution failed. reason: JSONThreatP
```

The square brackets ([]) and colons (:) in the regular expression must be escaped by using a backslash because they are reserved characters in regular expressions.

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

7. Insert the following step inside the **FaultRule** element:

```
<Step>
  <Condition>fault.name Matches "ExecutionFailed"
</Condition>
  <Name>AM-400JSONThreat</Name>
</Step>
```

content_co

The final FaultRules section in the ProxyEndpoint looks like this:

```
<FaultRules>
  <FaultRule name="JSONThreat">
    <Condition>jsonattack.JSONTP-Protect.failed == true</Condition>
    <Step>
      <Condition>fault.name Matches "ExecutionFailed"</Condition>
      <Name>AM-400JSONThreat</Name>
    </Step>
  </FaultRule>
</FaultRules>
```

8. Click **Save**, and then click **Save as New Revision**.

Task 2. Validate incoming requests

In this task you raise a fault if the request is not correct.

You may have noticed that if you do not pass a **Content-Type** header set to **application/json**, the **JSONThreatProtection** policy will not validate the request.

To prevent this scenario, you should raise a fault if the Content-Type header is required but not correctly specified.

Create a conditional RaiseFault policy

1. In the Navigator pane, click **Proxy endpoints > default > createOrder**.
2. In the **createOrder request flow**, click **Add Policy Step (+)**.
3. In the **Add policy step** pane, select **Create new policy**, and then select **Mediation > Raise Fault**.
4. Specify the following values:

Property	Value
Name	RF-InvalidContentType
Display name	RF-InvalidContentType
Condition	request.header.Content-Type != "application/json"

5. Click **Add**.
6. In the Navigator pane, click **Policies > RF-InvalidContentType**.
7. Change the *RF-InvalidContentType* policy configuration to:

```
<RaiseFault continueOnError="false" enabled="true" name="RF-InvalidContentType" content_cc="content_cc">
  <FaultResponse>
    <Set>
      <Headers/>
      <Payload contentType="application/json">{
        "error":"Content-Type header must be application/json"
      }</Payload>
    <StatusCode>400</StatusCode>
  </FaultResponse>
</RaiseFault>
```

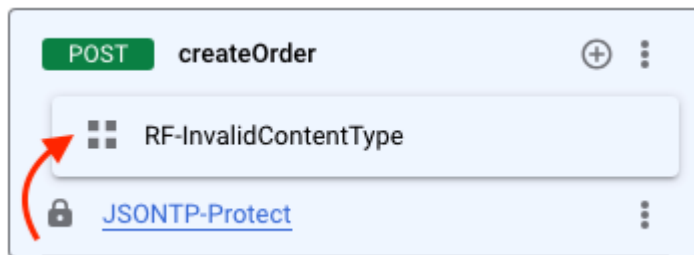
```

    <ReasonPhrase>Bad Request</ReasonPhrase>
  </Set>
</FaultResponse>
<IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
</RaiseFault>

```

Note: Because you are setting the desired response in this policy, you do not need to create a corresponding FaultRule.

8. In the Navigator pane, click **Proxy endpoints > default > createOrder**.
9. In the Request createOrder flow*, select the the RaiseFault policy and drag it so it precedes the JSONThreatProtection policy:



The *createOrder* flow XML should now look like this:

```

<Flow name="createOrder">
  <Description>Create a new order</Description>
  <Request>
    <Step>
      <Condition>request.header.Content-Type != "application/json"</Condi
      <Name>RF-InvalidContentType</Name>
    </Step>
    <Step>
      <Name>JSONTP-Protect</Name>
    </Step>
  </Request>
  <Response/>
  <Condition>(proxy.pathsuffix MatchesPath "/orders") and (request.verb = "
</Flow>

```

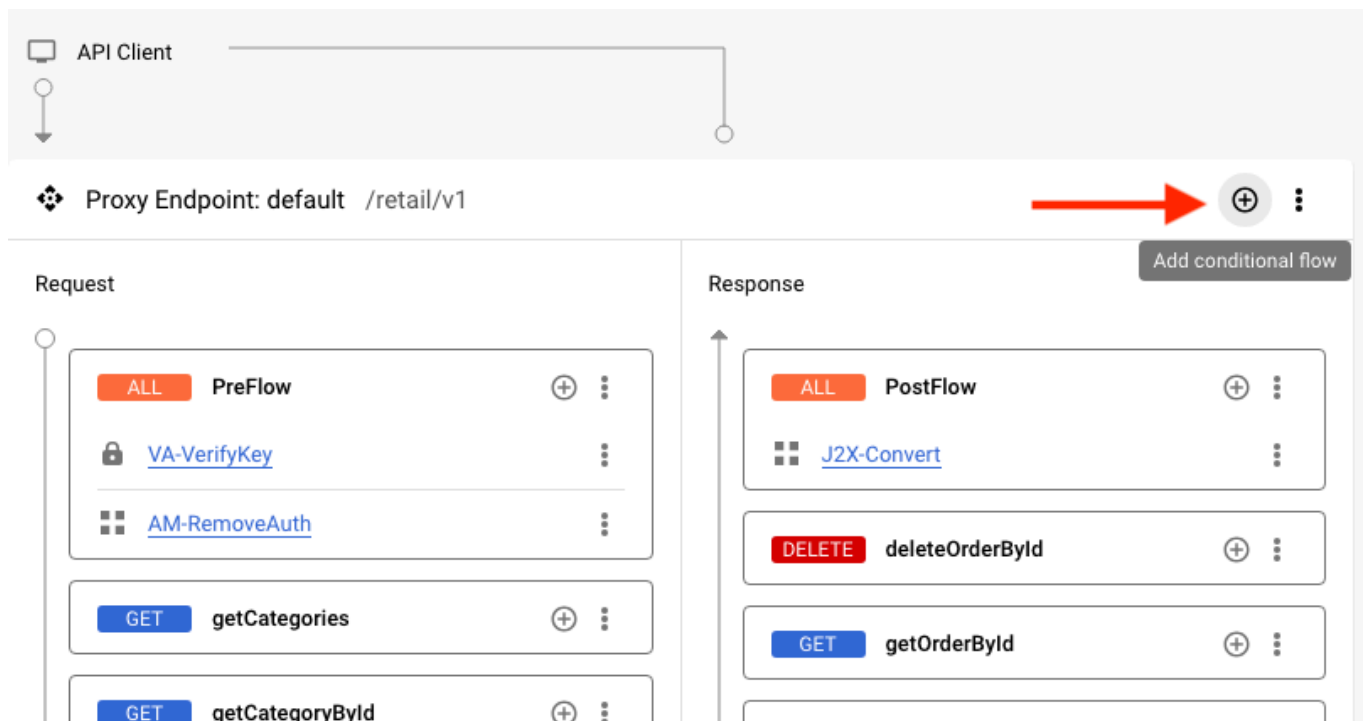
10. Click **Save**.

Task 3. Create a new default conditional flow to handle requests to invalid resources

In this task, you block all requests that don't match a conditional flow in the proxy endpoint request.

Add a new conditional flow

1. In the Navigator pane, click **Proxy endpoints > default**.
2. In the graphical representation of the *default* proxy endpoint, click **Add Conditional Flow (+)**:



3. Set the following properties. Because the wizard does not allow you to create a flow without a condition, you will remove the condition manually:

Property	Value
Flow Name	404NotFound
Description	Return 404 Not Found
Condition Type	select <i>Custom</i>
Condition	DELETE THIS

4. Click **Add**.

5. Remove the "DELETE THIS" condition from the 404NotFound flow. This flow should always be the *last* flow in the list of conditional flows. With no condition, the 404NotFound flow will always be executed if none of the previous flows are executed.

After you remove the "DELETE THIS" condition, your 404NotFound flow should look like this:

```
<Flow name="404NotFound">
  <Description>Return 404 Not Found</Description>
  <Request/>
  <Response/>
</Flow>
```

Add a new RaiseFault policy

1. In the Navigator pane, click **Proxy endpoints > default > 404NotFound**.
2. In the **404NotFound request flow**, click **Add Policy Step (+)**.
3. In the **Add policy step** pane, select **Create new policy**, and then select **Mediation > Raise Fault**.
4. Specify the following values:

Property	Value
Name	RF-404NotFound
Display name	RF-404NotFound

5. Click **Add**.
6. In the Navigator pane, click **Policies > RF-404NotFound**.
7. Change the *RF-404NotFound* policy configuration to:

```
<RaiseFault continueOnError="false" enabled="true" name="RF-404NotFound">
  <FaultResponse>
    <Set>
      <Headers/>
      <Payload contentType="application/json">{
        "error": "Invalid request: {request.verb} {proxy.basepath}
        {proxy.pathsuffix}"
      }</Payload>
      <StatusCode>404</StatusCode>
      <ReasonPhrase>Not Found</ReasonPhrase>
    </Set>
  </FaultResponse>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
</RaiseFault>
```

content_co

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

Check deployment status

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

The screenshot shows the 'Deployments' section of the Google Cloud Apigee console. A table lists the deployment status:

Status	Revision	Environment	Action
✓	1	eval	UNDEPLOY

A tooltip is displayed over the green checkmark status icon, showing:

- ✓ eval
- Status: Deployed
- Deployed on: 11/27/2023, 6:19:20 PM

Below the deployment table, there is a table for the proxy details:

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.

The screenshot shows the Google Cloud Apigee console interface. The left sidebar contains navigation options: Overview, Proxy development, API proxies (selected), Shared flows, Integrations, Offline debug, API monitor, Distribution, API product, Portals, and Developers.

The main content area shows the 'retail-v1' proxy configuration. The 'OVERVIEW' tab is active, displaying a 'Proxy summary' and a 'Deployments' section. The 'Deployments' section shows a table with a red warning icon and the text '1 Disruption'.

Status	Revision	Environment	Action
!	1	eval	UNDEPLOY

A tooltip is displayed over the red warning icon, showing:

- ! eval
- Status: no instances are reporting status for this environment
- Deployed on: 11/27/2023, 6:19:20 PM

Below the deployment table, there is a table for the proxy details:

Revision	Description	Last modified
1	My retail API	November 27, 2023

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.

Compute Engine

Virtual machines

VM instances

Instance templates

Sole-tenant nodes

Machine images

TPUs

Committed use discounts

VM instances

CREATE INSTANCE

IMPORT VM

REFRESH

INSTANCES

OBSERVABILITY

INSTANCE SCHEDULES

VM instances

Filter

Enter property name or value

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	✓	apigee-proxy-07m8	us-central1-f		apigee-proxy-group	10.128.0.3 (nic0)	34.27.73.218 (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	✓	apigee-proxy-dg85	us-central1-f		apigee-proxy-group	10.128.0.4 (nic0)	34.132.15.243 (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	✓	apigee-proxy-gdnx	us-central1-f		apigee-proxy-group	10.128.0.5 (nic0)	35.188.25.205 (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	✓	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

While you wait, learn more by reviewing:

- RaiseFault policy
- Handling faults

Task 4. Test the retail API

In this task, you validate the three changes you made to the proxy.

Store the app's key in a shell variable

The API key may be retrieved directly from the app accessible on the **Publish > Apps** page. It can also be retrieved via Apigee API call.

- In **Cloud Shell**, run the following command:

```
export API_KEY=$(curl -q -s -H "Authorization: Bearer $(gcloud auth print-access-token)" -X GET "https://apigee.googleapis.com/v1/organizations/${GOOGLE_CLOUD_PROJECT}app" | jq --raw-output '.credentials[0].consumerKey'); echo "export API_KEY=${API_KEY}" >> ~/.profile; echo "API_KEY=${API_KEY}"
```

This command retrieves a Google Cloud access token for the logged-in user, sending it as a Bearer token to the Apigee API call. It retrieves the **retail-app** app details as a JSON response, which is parsed by **jq** to retrieve the app's key. That key is then put into the **API_KEY** environment variable, and the export command is concatenated onto the **.profile** file which runs automatically when starting a Cloud Shell tab.

Note: If you run the command and it shows **API_KEY=null**, the runtime instance is probably not yet available.

Test the JSONThreatProtection policy error

When a **POST /orders** payload exceeds the limits specified in the **JSONTP-Protect** JSONThreatProtection policy, the error should now be simpler.

- In Cloud Shell, execute this curl command:


```
curl -H "Content-Type: application/json" -H "apikey: ${API_KEY}" -X
POST "https://api-
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/orders" -d \
'{
  "orderNumber": 342345,
  "lineItems": [
    { "productId": "ME089LLA", "quantity": 1 },
    { "productId": "MD388LLA", "quantity": 2 },
    { "productId": "ME761LLB", "quantity": 3 },
    { "productId": "MD878LLA", "quantity": 4 }
  ],
  "promisedDeliveryDate": "30 Oct 2020",
  "deliveryNotes": "If not home, please place inside backyard
gate",
  "destination": {
    "addressType": "home",
    "address": {
      "streetAddr1": "1 Main St."
    }
  }
}' | json_pp
```

This payload contains an array with four product objects, but only three array elements are allowed by the JSONTP-Protect policy.

If you rewrote the error message correctly, this clean error message should be displayed:

```
{
  "error": "Invalid JSON payload: Exceeded array element count at line 7"
}
```

Your fault rule is *not* successfully rewriting the error if your error looks like this:

```
{
  "fault": {
    "faultstring": "JSONThreatProtection[JSONTP-Protect]: Execution failed.
reason: JSONThreatProtection[JSONTP-Protect]: Exceeded array element count
at line 7",
```

```
"detail": {
  "errorcode": "steps.jsonthreatprotection.ExecutionFailed"
}
}
```

Note: Use the debug tool to debug any issues.

Test the incorrect content type error

When the Content-Type header is not application/json, an error should be returned.

- Test the call without a Content-Type header:

```
curl -H "apikey: ${API_KEY}" -X POST "https://api-
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/orders" -d \
'{
  "orderNumber": 342345,
  "lineItems": [
    { "productId": "ME089LLA", "quantity": 1 },
    { "productId": "MD388LLA", "quantity": 2 },
    { "productId": "ME761LLB", "quantity": 3 },
    { "productId": "MD878LLA", "quantity": 4 }
  ],
  "promisedDeliveryDate": "30 Oct 2020",
  "deliveryNotes": "If not home, please place inside backyard
gate",
  "destination": {
    "addressType": "home",
    "address": {
      "streetAddr1": "1 Main St."
    }
  }
}' | json_pp
```

content_type

The proxy should now return a **400 Bad Request** error, indicating that the Content-Type header must be application/json:

```
{
  "error": "Content-Type header must be application/json"
}
```

If you use the debug tool to debug the proxy, the result is that the RF-InvalidContentType RaiseFault policy returns the error, and the JSONThreatProtection policy does not run.

Note: Use the debug tool to debug any issues.

Test the 404 Not Found error

When a request does not match any existing conditional flows, a 404 Not Found error should be returned.

- Test an invalid API request:

```
curl -H "apikey: ${API_KEY}" -X GET "https://api-content_co
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/tests" |
json_pp
```

You should receive a **404** error:

```
{
  "error": "Invalid request: GET /retail/v1/tests"
}
```

Note: If you add new flows using the UI + (add flow) button, the new flow will automatically be added to the end of the list of conditional flows. You must always ensure that the 404 flow is the last flow listed in the conditional flows.

Congratulations!

In this lab, you learned how to handle faults and manually raise faults.

End your lab