



Aug, 2015


Foundational Training

Fault Handling



Apigee Fault Handling

```
<Step>
  <Name>SpikeArrest</Name>
</Step>
<Step>
  <Name>SetConfigurationVariables</Name>
</Step>
<Step>
  <Name>VerifyApiKey</Name>
</Step>
<Step>
  <Name>QuotaPolicy</Name>
</Step>
```



Faults in Apigee are similar to exceptions in programming languages. When a fault is raised, current policy processing is aborted and switches to error processing called **<FaultRules>**. This is supported in **ProxyEndpoint** and **TargetEndpoint** configurations only.

```
1 <DefaultFaultRule>
2   <Step>
3     <Name>SyslogPolicy</Name>
4   </Step>
5   <AlwaysEnforce>true</AlwaysEnforce>
6 </DefaultFaultRule>
7 <FaultRules>
8   <FaultRule name = "Fault.InvalidKey">
9     <Step>
10      <Name>Add-WWW-Authenticate-Header</Name>
11    </Step>
12    <Condition>(fault.name == "invalid_consumer_key")</Condition>
13  </FaultRule>
14 </FaultRules>
15 <HTTPProxyConnection>
16   <BasePath>/certification/v1/weather</BasePath>
17   <VirtualHost>default</VirtualHost>
18   <VirtualHost>secure</VirtualHost>
19 </HTTPProxyConnection>
```

Apigee Fault Handling (cont'd)

<FaultRule>

- Support for multiple fault rules, executed conditionally
- Raised manually or automatically upon policy failure
- If multiple conditional FaultRules defined, their conditions are evaluated in reverse order (bottom up).



```
1  <DefaultFaultRule>
   <Step>
     <Name>SyslogPolicy</Name>
   </Step>
   <AlwaysEnforce>true</AlwaysEnforce>
 </DefaultFaultRule>
14 <FaultRules>
   <FaultRule name ="Fault.InvalidKey">
     <Step>
       <Name>Add-WWW-Authenticate-Header</Name>
     </Step>
     <Condition>(fault.name == "invalid_consumer_key")</Condition>
   </FaultRule>
15 </FaultRules>
16 <HTTPProxyConnection>
17   <BasePath>/certification/v1/weather</BasePath>
18   <VirtualHost>default</VirtualHost>
19   <VirtualHost>secure</VirtualHost>
   </HTTPProxyConnection>
```

<DefaultFaultRule>

- A catch-all / post processing fault rule is available using the flow
- If no FaultRule is matched, this flow will execute.
- Use <AlwaysEnforce>true</AlwaysEnforce> to use this flow for post processing in error flows. This will ensure the flow executes after a matched fault rule has completed its processing.

3 scenarios where processing will switch to fault processing:

- Using RaiseFault policy
- Any policy failure when continueOnError=false (default setting)
- Non-success response received from service callout or backend request (4XX, 5XX status codes).

Rewriting Backend Error Responses

Apigee Edge has a pre-defined fault response format:

```
1  {
2      "fault": {
3          "faultstring": "%errorMessage#",
4          "detail": {
5              "errorcode": "%errorCode#"
6          }
7      }
8  }
9
```

You can either:

1. Rewrite backend error responses to match Apigee's format
2. Rewrite Apigee fault responses to match the backend error response format

Day 3 – Lab 11 Fault Rules

- Create a fault rule in Proxy Endpoint

```
<FaultRules>
  <FaultRule name="SpikeArrestViolation">
    <Condition>fault.name == "SpikeArrestViolation"</Condition>
    <Step><Name>Spike-Arrest-Violation</Name></Step>
  </FaultRule>
</FaultRules>
```

- Create a Assign Message Policy named “Spike-Arrest-Violation”

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AssignMessage async="false" continueOnError="false" enabled="true" name="Spike-Arrest-Violation">
  <DisplayName>Spike Arrest Violation</DisplayName>
  <FaultRules/>
  <Properties/>
  <Set>
    <Headers/>
    <Payload contentType="text/plain">{"error":"Spike Arrest Voilation. Please ensure you are within the spike arrest voilation limits."} </Payload>
    <StatusCode>401</StatusCode>
    <ReasonPhrase>Unauthorized</ReasonPhrase>
  </Set>
  <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
  <AssignTo createNew="false" transport="http" type="request"/>
</AssignMessage>
```

- Test your proxy using Postman

Thank you

A network diagram consisting of white circular nodes of varying sizes connected by thin white lines. The nodes are scattered across the red background, with some forming a central cluster and others positioned more peripherally. The lines connect the nodes in a non-uniform, web-like structure.

apigee

March 2014