Fault Rules and Error Responses

Google Cloud

Terminology

1) Raise Fault policy

Terminology

1) Raise Fault policy

2) Apigee Edge raises an error during policy execution

Terminology

1) Raise Fault policy

2) Apigee Edge raises an error during policy execution

3) Fault Rule = Fault Handler (error flow)

Fault Handling

<FaultRules> located in

- **Proxy Endpoint**
- **Target Endpoint**

Fault Handling

<FaultRules> located in

- **Proxy Endpoint**
- **Target Endpoint**

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

apigee

Fault Handling

<FaultRules> located in

- **Proxy Endpoint**
- **Target Endpoint**

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

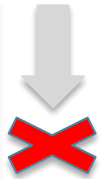
apigee

Fault Handling

<FaultRules> located in

- **Proxy Endpoint**
- **Target Endpoint**

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



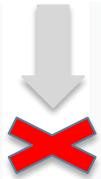
apigee

Fault Handling

<FaultRules> located in

- **Proxy Endpoint**
- **Target Endpoint**

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



```
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

<FaultRules> located in

- **Proxy Endpoint**
- **Target Endpoint**

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

```
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

Proxy Endpoint

```
<DefaultFaultRule>
    <Step>
        <Name>SyslogPolicy</Name>
    </Step>
    <AlwaysEnforce>true</AlwaysEnforce>
</DefaultFaultRule>
<FaultRules>
    <FaultRule name ="Fault.InvalidKey">
        <Step>
            <Name>Add-WWW-Authenticate-Header</Name>
        </Step>
    <Condition>(fault.name =="invalid_consumer_key")</Condition>
    </FaultRule>
</FaultRules>
```

Fault Handling

Proxy Endpoint

```
<DefaultFaultRule>
    <Step>
        <Name>SyslogPolicy</Name>
    </Step>
    <AlwaysEnforce>true</AlwaysEnforce>
</DefaultFaultRule>
<FaultRules>
    <FaultRule name ="Fault.InvalidKey">
        <Step>
            <Name>Add-WWW-Authenticate-Header</Name>
        </Step>
    <Condition>(fault.name =="invalid_consumer_key")</Condition>
    </FaultRule>
</FaultRules>
```

Fault Handling

Proxy Endpoint

```
<DefaultFaultRule>
    <Step>
        <Name>SyslogPolicy</Name>
    </Step>
    <AlwaysEnforce>true</AlwaysEnforce>
</DefaultFaultRule>
<FaultRules>
    <FaultRule name ="Fault.InvalidKey">
        <Step>
            <Name>Add-WWW-Authenticate-Header</Name>
        </Step>
    <Condition>(fault.name =="invalid_consumer_key")</Condition>
    </FaultRule>
</FaultRules>
```

Target Endpoint

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

apigee

Fault Handling

Proxy Endpoint

```
<DefaultFaultRule>
    <Step>
        <Name>SyslogPolicy</Name>
    </Step>
    <AlwaysEnforce>true</AlwaysEnforce>
</DefaultFaultRule>
<FaultRules>
    <FaultRule name ="Fault.InvalidKey">
        <Step>
            <Name>Add-WWW-Authenticate-Header</Name>
        </Step>
    <Condition>(fault.name =="invalid_consumer_key")</Condition>
    </FaultRule>
</FaultRules>
```

Target Endpoint

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

Fault Handling

- <FaultRules>
  - Support for multiple fault rules, executed conditionally.

Fault Handling

- <FaultRules>
  - Support for multiple fault rules, executed conditionally.
- <DefaultFaultRule>
  - catch all
  - executes when no FaultRule matches
  - <AlwaysEnforce>

Rewriting Backend Error Responses

Edge has a pre-defined fault response format:

```
{
   "fault":{
      "faultstring":"%errorMessage#",
      "detail":{
         "errorcode":"%errocode#"
      }
   }
}
```

Rewriting Backend Error Responses

Edge has a pre-defined fault response format:

```
{
   "fault":{
      "faultstring":"%errorMessage#",
      "detail":{
         "errorcode":"%errocode#"
      }
   }
}
```

You can either:

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

Raise Fault policy

Code: raise_fault_invalid_post

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RaiseFault async="false" continueOnError="false" enabled="true" name="raise_fault_invalid_post">
    <DisplayName>raise_fault_invalid_post</DisplayName>
    <FaultRules/>
    <Properties/>
    <FaultResponse>
        <Set>
            <Headers/>
            <Payload contentType="application/json">\{"error":"Invalid Post Data"}</Payload>
            <StatusCode>400</StatusCode>
            <ReasonPhrase>Bad Request</ReasonPhrase>
        </Set>
    </FaultResponse>
    <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
</RaiseFault>
```

apigee

Raise Fault policy

```
<PreFlow name="PreFlow">
       <Request>
           <Step>
               <Name>RaiseFaultMissingNameParam</Name>
               <Condition>request.queryparam.name == null</Condition>
           </Step>
       </Request>
       <Response/>
   </PreFlow>
```

Thank You

Google Cloud

Fault Handling

<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).

<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 contineOnError=false (default setting).
- Non-success response received from service callout or

Raise Faults work with conditions

Select Post Ratings Resource from navigation pane

Add a condition to the Step (policy)

Code: post_ratings

```
1
2         <Flow name="post_ratings">
3             <Description/>
4             <Request>
5                 <Step>
6                     <FaultRules/>
7                     <Name>extract_post_data_ratings</Name>
8                 </Step>
9                 <Step>
10                    <FaultRules/>
11                    <Name>raise_fault_invalid_post</Name>
12                    <Condition>((truck = NULL) or (comment = NULL) or (commenter = NULL) or (score = NULL))</Condition>
13                </Step>
14            </Request>
15            <Response/>
16            <Condition>(proxy.pathsuf  latchesPath &quo  tings&quot;) an  quest.verb = &q  OST&quot;)</Condition>
17        </Flow>
```

Raise Fault policy

Add a Raise Fault Policy

Name and Select the Proxy Endpoint (Ratings (Post)) from the flows,

Modify the content to return an appropriate fault message

Code: raise_fault_invalid_post

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RaiseFault async="false" continueOnError="false" enabled="true" name="raise_fault_invalid_post">
    <DisplayName>raise_fault_invalid_post</DisplayName>
    <FaultRules/>
    <Properties/>
    <FaultResponse>
        <Set>
            <Headers/>
            <Payload contentType="application/json">\{"error":"Invalid Post Data"}</Payload>
            <StatusCode>400</StatusCode>
            <ReasonPhrase>Bad Request</ReasonPhrase>
        </Set>
    </FaultResponse>
    <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
</RaiseFault>
```

Raise Fault Policy Details

Fault Response Tag Format

- Set – Allows you to build an inline response to fault
  - StatusCode – HTTP Status Code
  - ReasonPhrase – HTTP Reason Phrase
  - Payload – Message Contents for your fault (follows same spec as assign message)
- Copy – Allows you to copy the request or the response into the fault response
  - Attr:source – request or response object

Raise Faults work with conditions

Code: post_ratings

```
1
2       <Flow name="post_ratings">
3           <Description/>
4           <Request>
5               <Step>
6                   <FaultRules/>
7                   <Name>extract_post_data_ratings</Name>
8               </Step>
9               <Step>
10                  <FaultRules/>
11                  <Name>raise_fault_invalid_post</Name>
12                  <Condition>((truck = NULL) or (comment = NULL) or (commenter = NULL) or (score = NULL))</Condition>
13              </Step>
14          </Request>
15          <Response/>
16          <Condition>(proxy.pathsuffix MatchesPath &quot;/ratings&quot;) and (request.verb = &quot;POST&quot;)</Condition>
17      </Flow>
```