Google Cloud

Edge Fundamentals

Anatomy of an Edge Proxy

Anatomy of API Proxy

- Discussion
  - Explore the main logical components of the EDGE system from an Administrator's perspective
- Review Platform Entities
  - Organizations, Environments and Proxies

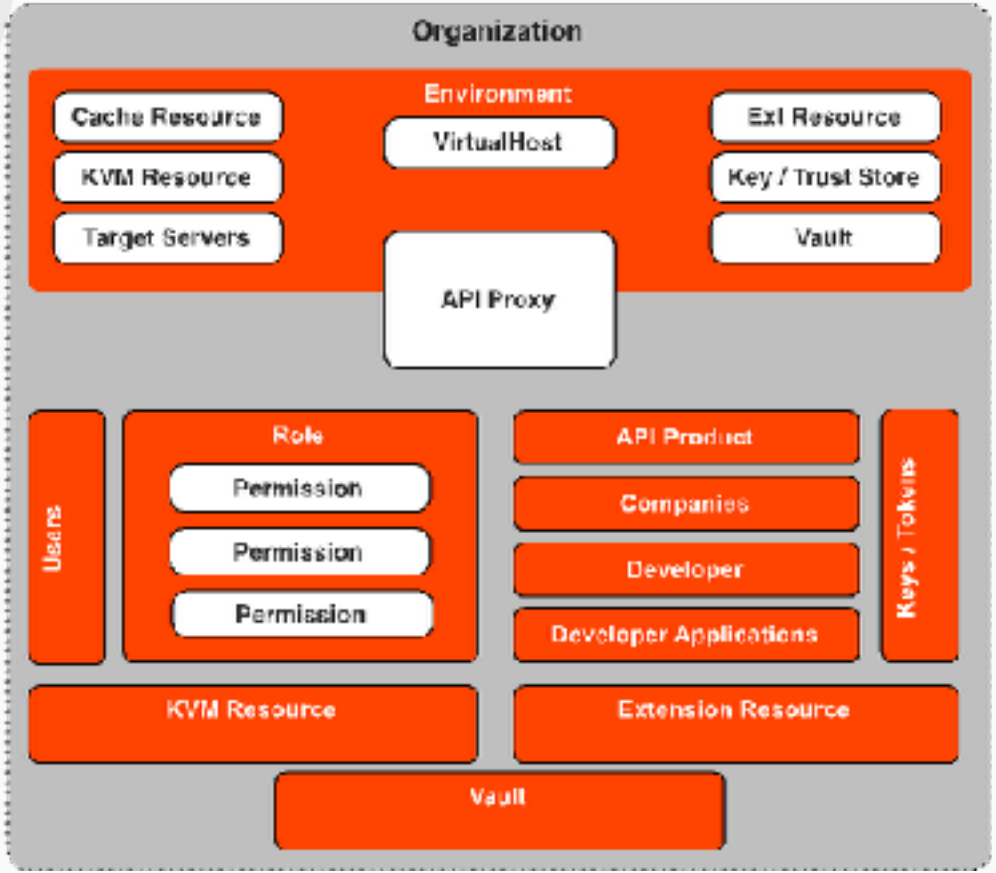
Proxy Endpoint

- FaultRules
- Virtual Hosts
- Flows
  - PreFlow
  - Conditional Flow
  - PostFlow
- Connection
- RouteRules

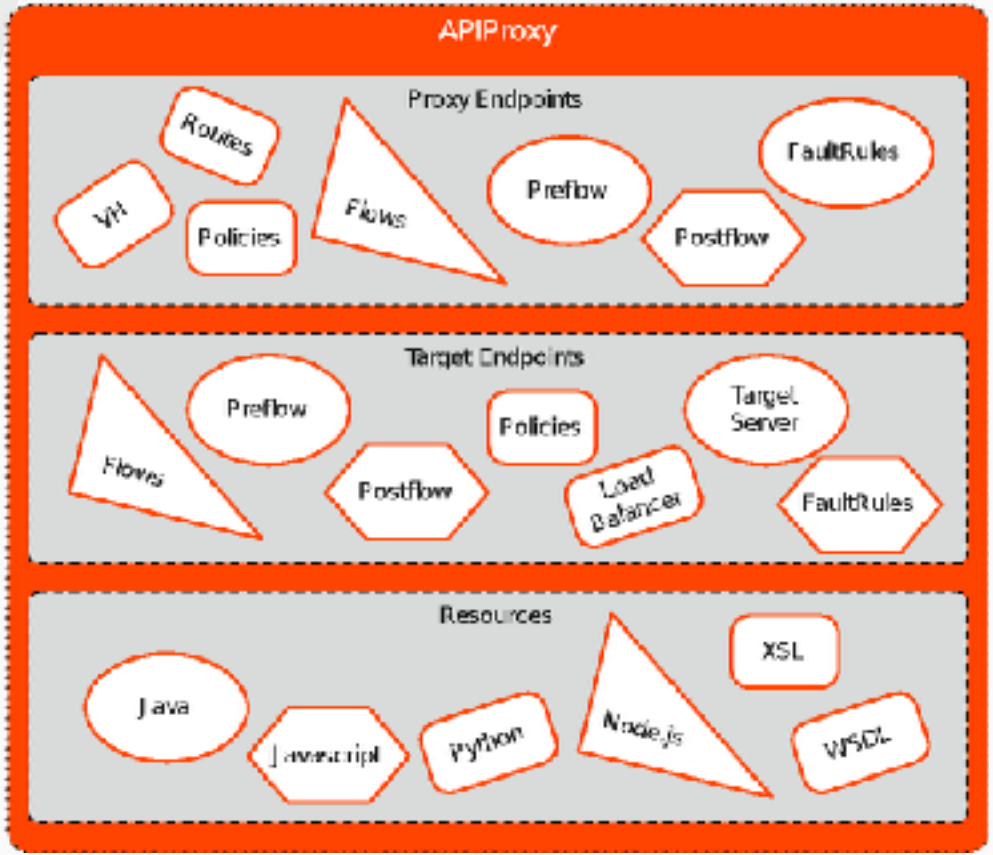
Target Endpoint

- FaultRules
- Flows
  - PreFlow
  - Conditional Flow
  - PostFlow
- Connection
- LoadBalancer

Platform Entities



API Proxies

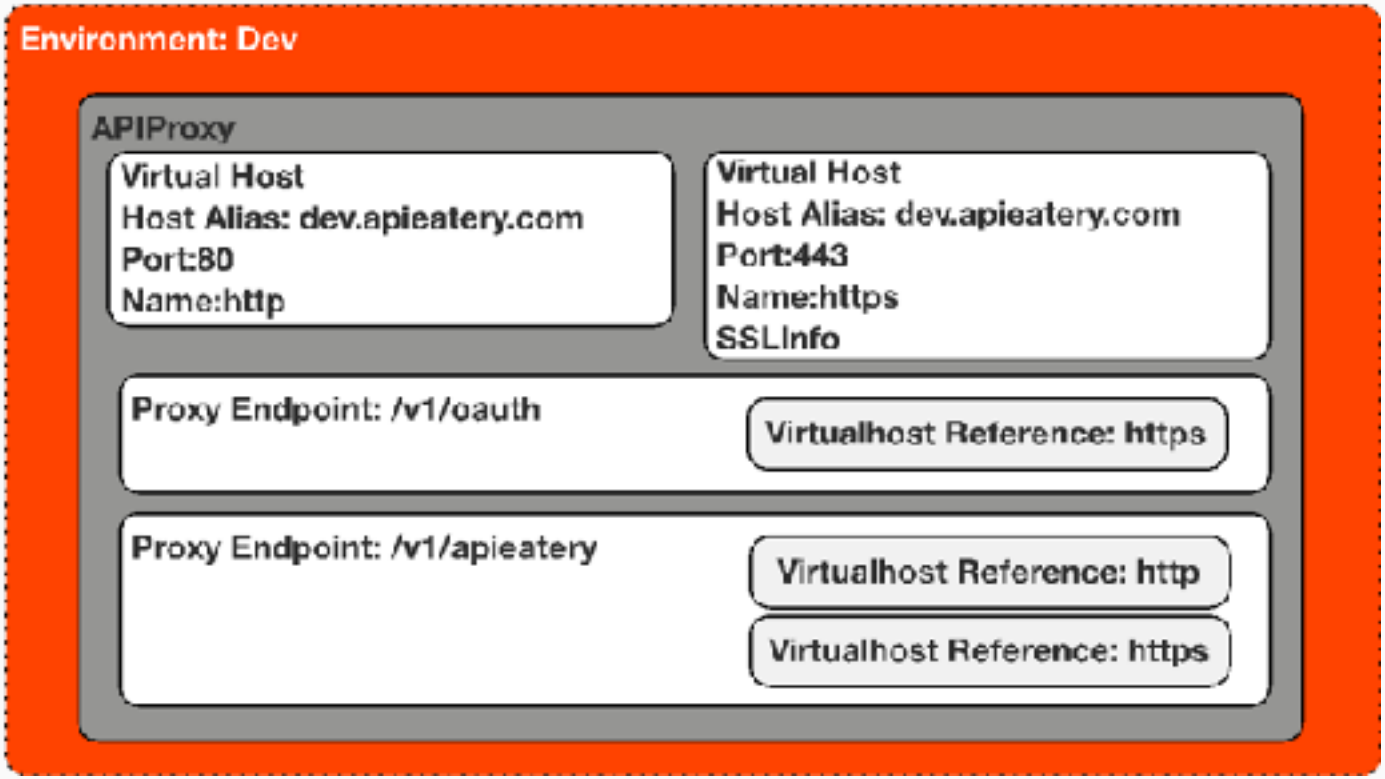


Proxy Endpoint

Proxy Endpoint Example

```
<ProxyEndpoint name="default">
    <DefaultFaultRule name="generic_fault_handler">
        <Step><Name>fault_genericfault</Name></Step>
        <AlwaysEnforce>true</AlwaysEnforce>
    </DefaultFaultRule>
    <FaultRules>
        <FaultRule name="invalid_key_rule">
            <Step><Name>fault_invalidkey</Name></Step>
            <Condition>fault.name = "InvalidApiKey"</Condition>
        </FaultRule>
    </FaultRules>
    <Flows/>
    <PreFlow name="PreFlow"/>
    <PostFlow name="PostFlow"/>
    <HTTPProxyConnection>
        <BasePath>/v1/someAPI</BasePath>
        <VirtualHost>default</VirtualHost>
    </HTTPProxyConnection>
    <RouteRule name="default">
        <TargetEndpoint>default</TargetEndpoint>
    </RouteRule>
</ProxyEndpoint>
```

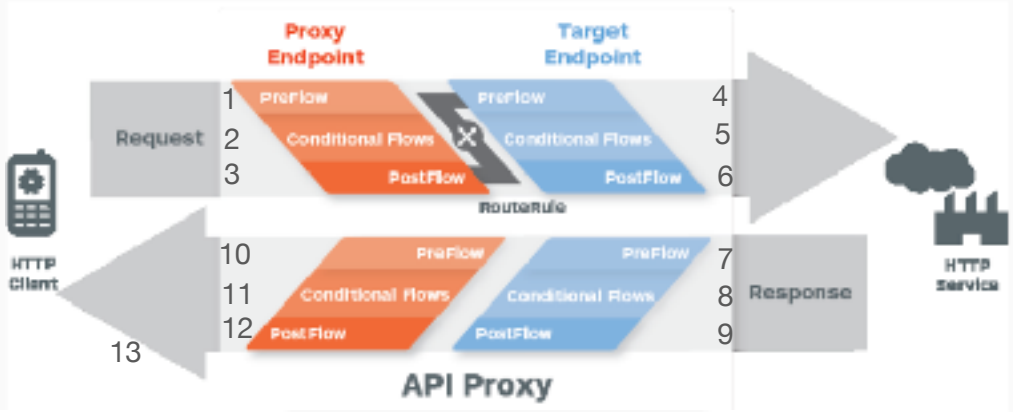
Virtual Hosts



Virtual Hosts

```
1. <HTTPProxyConnection>
2.     <BasePath>/v1/someAPI</BasePath>
3.     <VirtualHost>default</VirtualHost>
4.     <VirtualHost>secure</VirtualHost>
5.     <VirtualHost>beta</VirtualHost>
6. </HTTPProxyConnection>
```

Policy Flows



apiproxy/proxies/default.xml

```
<ProxyEndpoint name="default">
  <PreFlow>
    <Request>1</Request>
    <Response>10</Response>
  </PreFlow>
  <Flows>
    <Flow name="getDogs">
      <Request>2</Request>
      <Response>11</Response>
      <Condition></Condition>
    </Flow>
  </Flows>
  <PostFlow>
    <Request>3</Request>
    <Response>12</Response>
  </PostFlow>
  ...
```

apiproxy/targets/default.xml

```
<TargetEndpoint name="default">
  <PreFlow>
    <Request>4</Request>
    <Response>7</Response>
  </PreFlow>
  <Flows>
    <Flow name="purchaseTarget">
      <Request>5</Request>
      <Response>8</Response>
      <Condition></Condition>
    </Flow>
  </Flows>
  <PostFlow>
    <Request>6</Request>
    <Response>9</Response>
  </PostFlow>
  ...
```

Policy Flows

Global Flows

```
<PreFlow name="PreFlow">
    <Request>
        <Step><Name>Verify-Api-Key</Name></Step>
        <Step>
            <Name>Error-header</Name>
            <Condition>request.header.x-error = true</Condition>
        </Step>
    </Request>
    <Response/>
</PreFlow>
<PostFlow name="PostFlow">
    <Request/>
    <Response>
         <Step><Name>Custom-Analytics</Name></Step>
    </Response>
</PostFlow>
```

Policy Flows

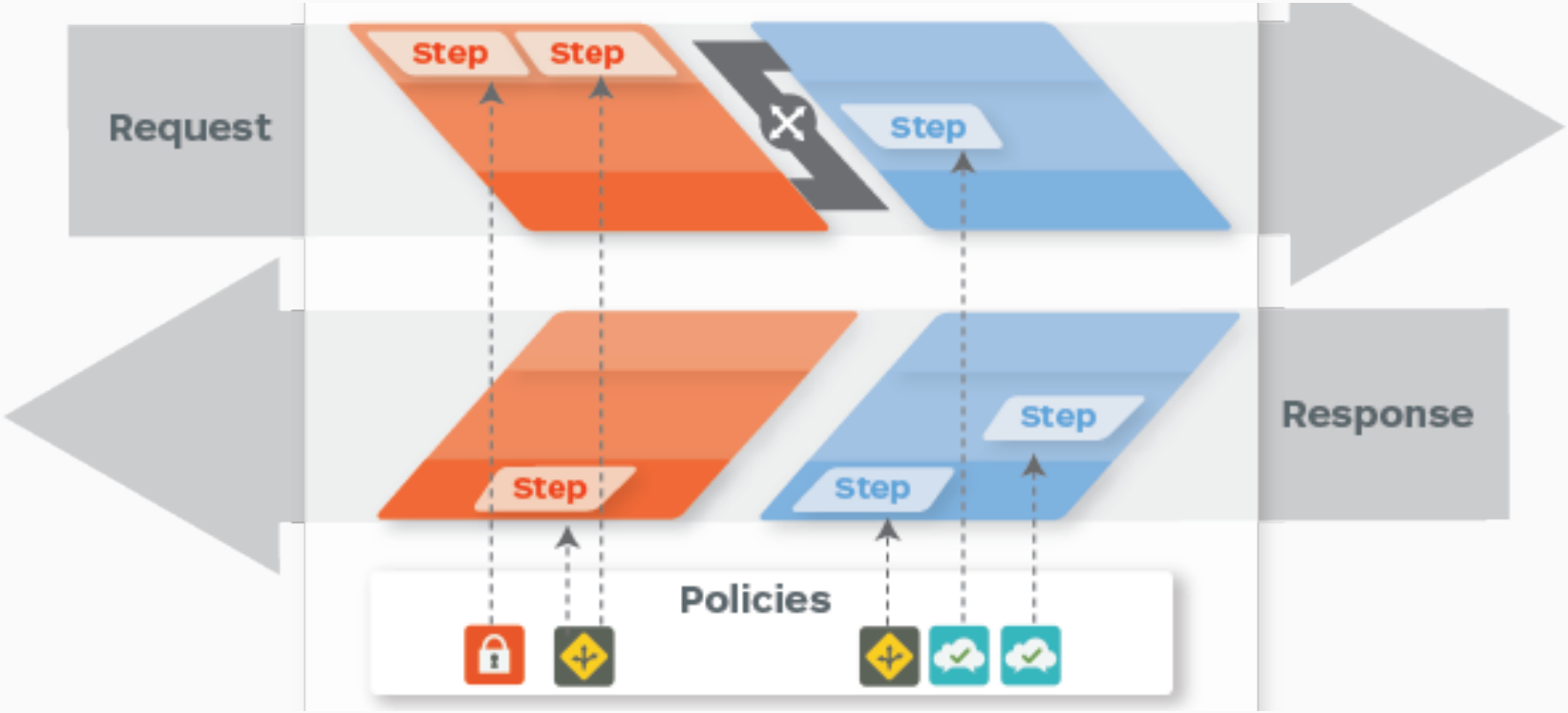
Conditional Flows / Resource

```
<Flows>
    <Flow name="GetResource">
        <Description>Get Resource</Description>
        <Request>
            <Step><Name>Resource-Quota</Name></Step>
        </Request>
        <Response/>
        <Condition>(proxy.pathsuffix MatchesPath "resource") and (request.verb = "GET")</Condition>
    </Flow>
</Flows>
```

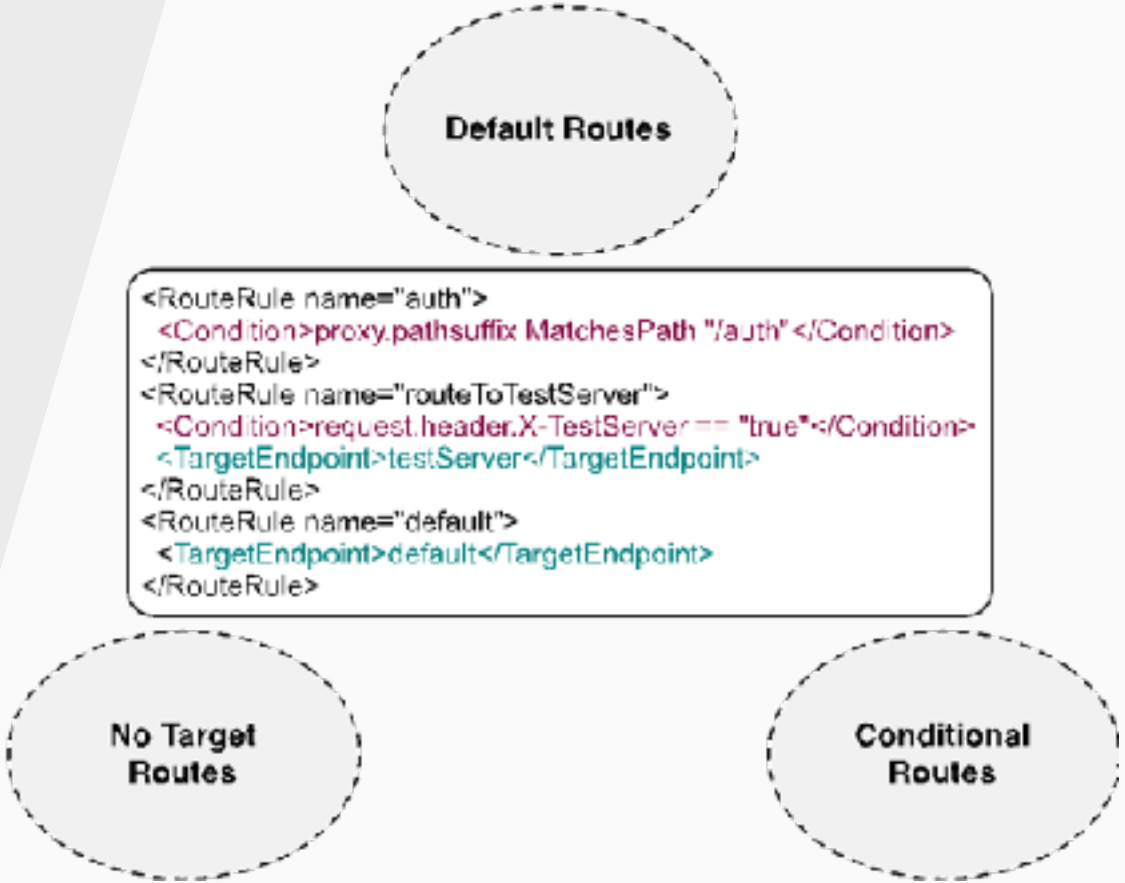
Post Client

```
<PostClientFlow>
    <Response>
        <Step><Name>Send-Audit-Info</Name></Step>
    </Response>
</PostClientFlow>
```

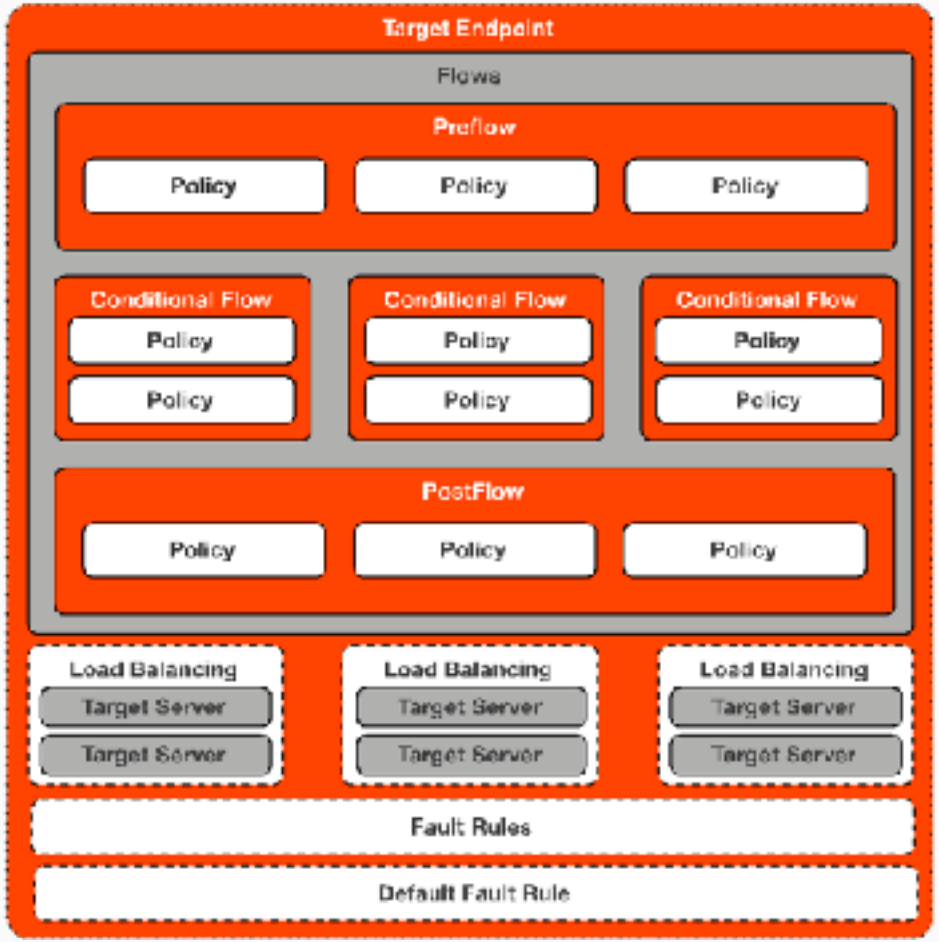
Policies



Route Rules



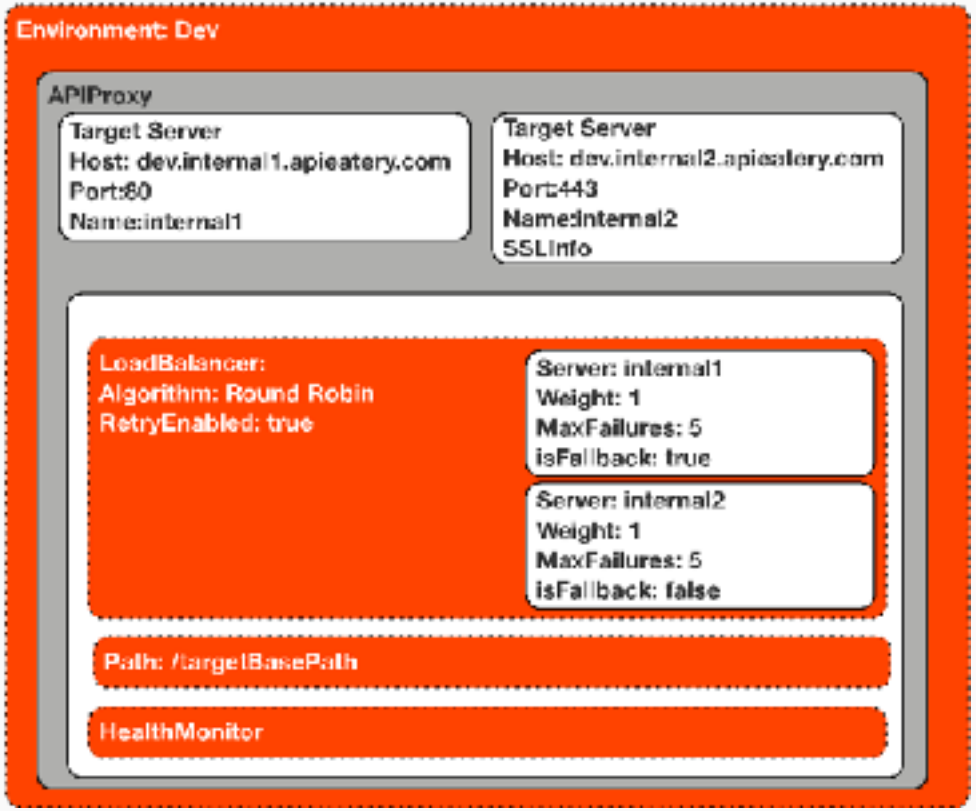
Target Endpoint



Target Endpoint Example

```
<TargetEndpoint name="default">
    <DefaultFaultRule name="generic_fault_handler">
        <Step><Name>fault_genericfault</Name></Step>
        <AlwaysEnforce>true</AlwaysEnforce>
    </DefaultFaultRule>
    <FaultRules>
        <FaultRule name="catch_all">
            <Step><Name>fault_genericfault</Name></Step>
        </FaultRule>
    </FaultRules>
    <PreFlow name="PreFlow">
        <Request>
            <Step><Name>set-target-url</Name></Step>
        </Request>
        <Response>
            <Step><Name>xml-to-json</Name></Step>
        </Response>
    </PreFlow>
    <Flows/>
    <PostFlow name="PostFlow"/>
    <HTTPTargetConnection>
        <URL>http://wsf.cdyne.com/WeetherWS/Weether.asmx</URL>
    </HTTPTargetConnection>
</TargetEndpoint>
```

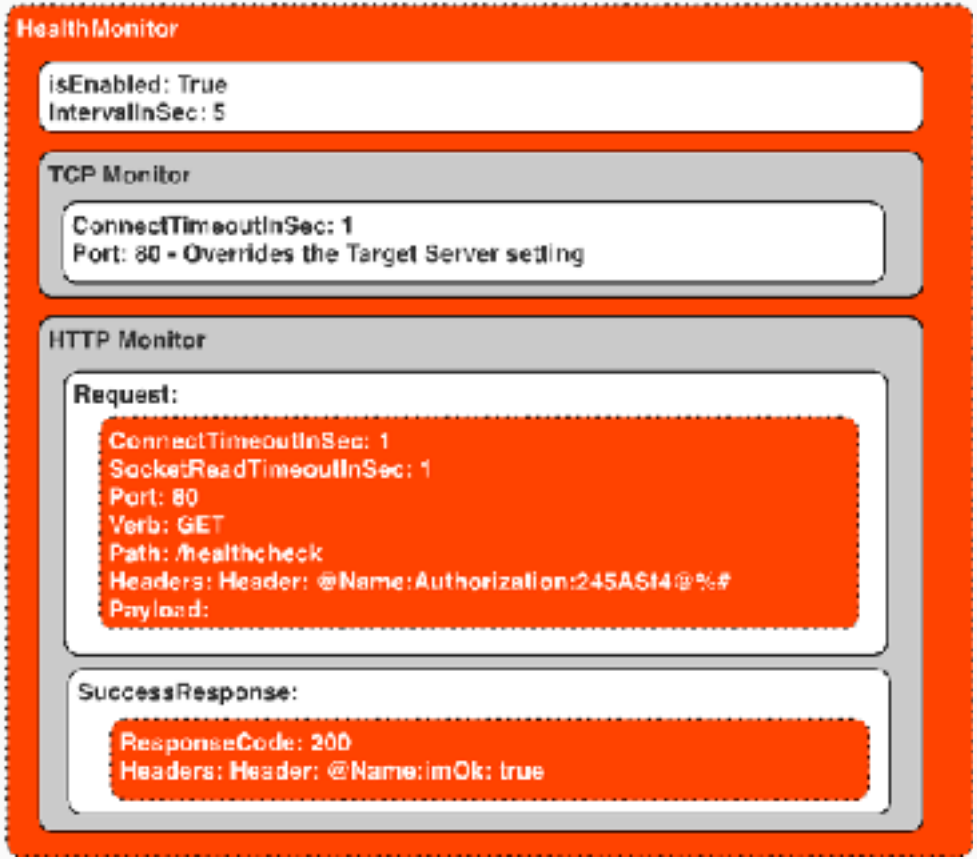
Load Balancing



Load Balancer Example

```
<HTTPTargetConnection>
    <LoadBalancer>
        <RetryEnabled>true</RetryEnabled>
        <Algorithm>Weighted</Algorithm>
        <Server name="server1">
            <MaxFailures>5</MaxFailures>
            <IsEnabled>true</IsEnabled>
            <IsFallback>false</IsFallback>
            <Weight>100</Weight>
        </Server>
        <Server name="server2">
            <MaxFailures>5</MaxFailures>
            <IsEnabled>true</IsEnabled>
            <IsFallback>true</IsFallback>
            <Weight>1</Weight>
        </Server>
    </LoadBalancer>
    <Path>/target/base/path</Path>
</HTTPTargetConnection>
```

Health Monitor



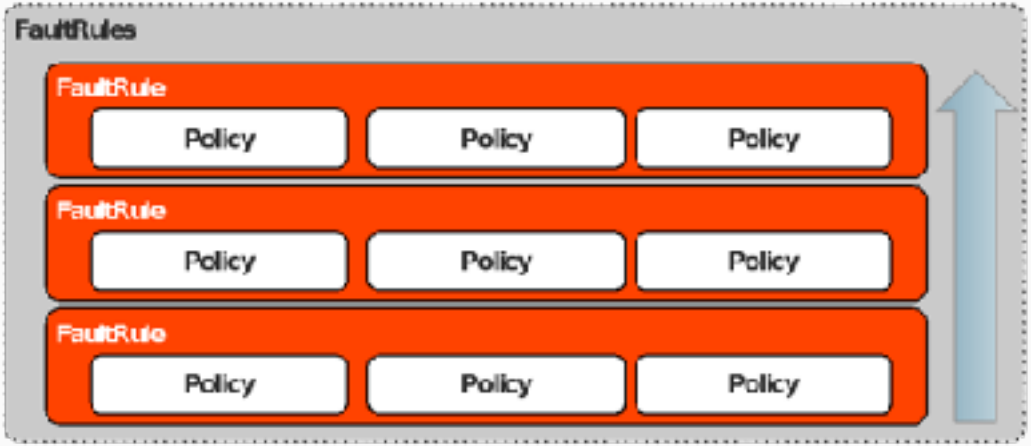
TCP Health Checks

```
1.  <HTTPTargetConnection>
2.  <LoadBalancer>
3.      <RetryEnabled>true</RetryEnabled>
4.      <Server name="server1"/>
5.      <Server name="server2"/>
6.  </LoadBalancer>
7.  <Path>/target/base/path</Path>
8.  <HealthMonitor>
9.      <IsEnabled>true</IsEnabled>
10.     <IntervalInSec>5</IntervalInSec>
11.     <TCPMonitor>
12.       <ConnectTimeoutInSec>1</ConnectTimeoutInSec>
13.     </TCPMonitor>
14. </HealthMonitor>
15. </HTTPTargetConnection>
```

HTTP Health Checks

```
<HTTPTargetConnection>
  <LoadBalancer>
      <RetryEnabled>true</RetryEnabled>
      <Server name="server1"/>
      <Server name="server2"/>
  </LoadBalancer>
  <Path>/target/base/path</Path>
  <HealthMonitor>
    <HTTPMonitor>
      <Request>
        <ConnectTimeoutInSec>10</ConnectTimeoutInSec>
        <SocketReadTimeoutInSec>30</SocketReadTimeoutInSec>
        <Port>80</Port>
        <Verb>GET</Verb>
        <Path>/healthcheck</Path>
        <Headers><Header name="X-Ping">ABC123</Header></Headers>
      </Request>
      <SuccessResponse>
        <ResponseCode>200</ResponseCode>
        <Headers><Header name="ImOK">YoureOK</Header></Headers>
      </SuccessResponse>
    </HTTPMonitor>
  </HealthMonitor>
</HTTPTargetConnection>
```

Fault Rules



Fault Rules

```
<DefaultFaultRule name="generic_fault_handler">
    <Step><Name>fault_genericfault</Name></Step>
    <AlwaysEnforce>true</AlwaysEnforce>
</DefaultFaultRule>
<FaultRules>
    <FaultRule name="catch_all">
        <Step><Name>fault_genericfault</Name></Step>
    </FaultRule>
    <FaultRule name="missing_key_rule">
        <Step><Name>fault_missingkey</Name></Step>
        <Condition>fault.name = "MissingApiKey"</Condition>
    </FaultRule>
    <FaultRule name="invalid_key_rule">
        <Step><Name>fault_invalidkey</Name></Step>
        <Condition>fault.name = "InvalidApiKey"</Condition>
</FaultRule>
</FaultRules>
```

Advanced Endpoint Properties For Proxy

In addition to the standard endpoint configurations there are some properties to control advanced functionality.

<HTTPProxyConnection>

- **request.streaming.enabled** – if the payload will not be read or updated during the proxy processing, you can enable to to avoid payload buffer size limits
- **response.streaming.enabled** - if the payload will not be read or updated during the proxy processing, you can enable to to avoid payload buffer size limits

Advanced Endpoint Properties For Target

<HTTPTargetConnection>

- **connect.timeout.millis** – this is used to set the connection timeout for backend connections (in ms). Defaults to 60000 (1 min) and highly recommended to lower
- **io.timeout.millis** – this is used to set the response read timeout (waiting for response from backend). Defaults to 120000 (2 min) and extremely encouraged to reduce to optimize connection handling in the event that a backend service is degraded.
- **request.streaming.enabled** - if the payload will not be read or updated during the target processing, you can enable to avoid payload buffer size limits
- **response.streaming.enabled** - if the payload will not be read or updated during the target processing, you can enable to to avoid payload buffer size limits
- **success.codes** – this property is used to set the response status codes that are treated as ‘success’. Defaults to 1XX,2XX,3XX. This is a helpful configuration when you want to access 4XX responses are success to continue response flow processing

THANK YOU