Google Cloud

Edge

Logging

Proprietary + Confidential

API Traffic Logging

- Are you using logging system at the moment?
  - **Splunk, fallback BaaS**
- Is it available externally? **Eventually**
- Any specific security requirements for sending information to logging servers?
  - Don’t log PCI/HIPAA
- How are you using logs / planning on using logs?
- What information has to captured in the logs?
  - Is it already present in the API transactions? Yes
- Is fire & forget logging sufficient or successful log event has to be verified during API runtime?
- Full log traceability between systems?
  - Correlation id for all API requests? Yes via messageid
  - How is this being created / managed if required?

Edge – Logging

Edge does not log any API traffic by default

- Its up to the API team to configure logging

Options for logging

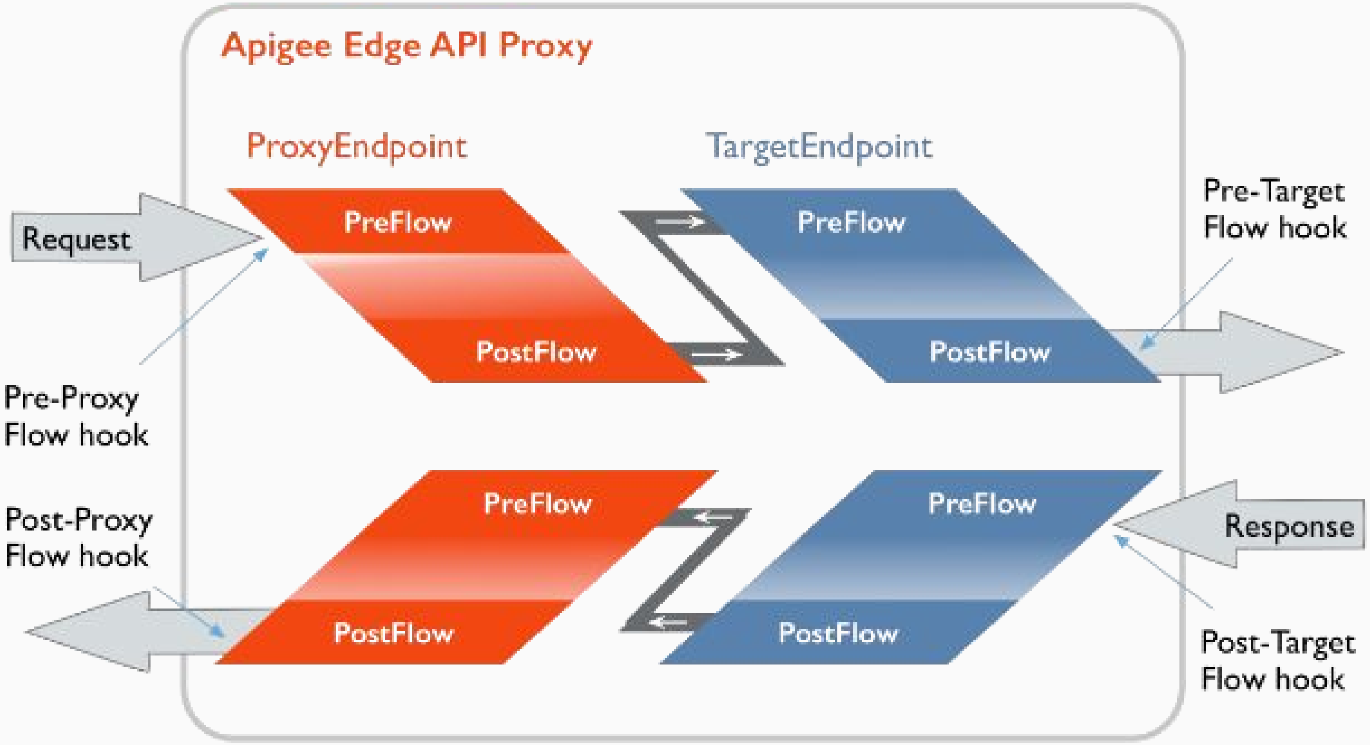
- Message Logging policy - Post Client Flow
  - Syslog
    - UDP – faster, *but not guaranteed*
    - TCP – guaranteed and supports TLS
- JavaScript Callout - not in Post Client Flow
  - Using async HTTP client (e.g. Sumo Logic)

Edge – Example Fields for Logging

- Organization name
- Environment name
- API name, version (or API proxy name)
- API Product name
- developer application name
- full resource URL including query string parameters
- request verb
- content length
- request received timestamp
- target request timestamp
- target response timestamp
- response sent timestamp
- total time in ms in Edge
- total time in ms in target
- total time for the whole response
- response code
- response message
- error message
- error code

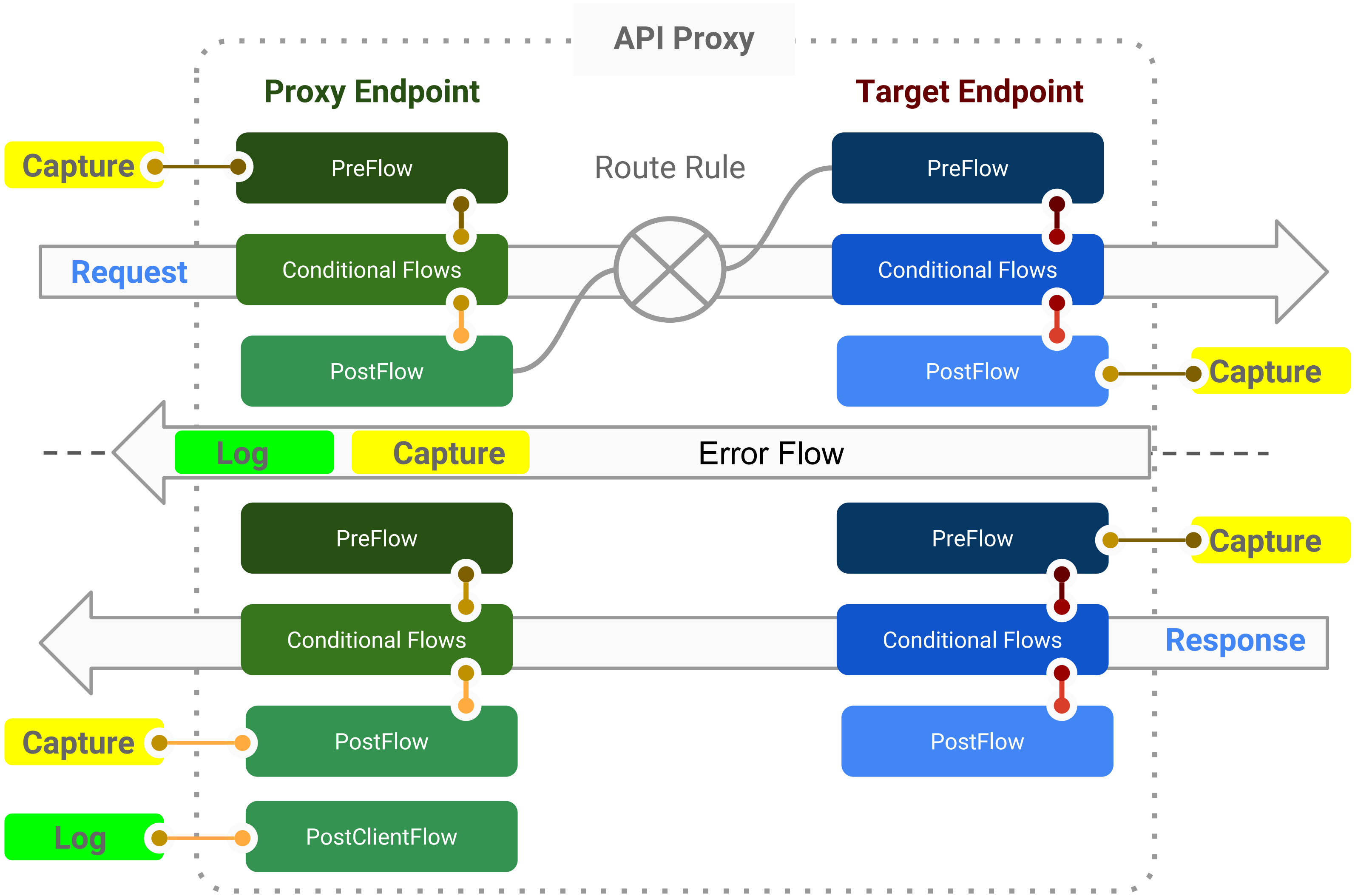
Edge – Where to Add Message Logging Policy

- Right after request comes in, right after response goes out (PostClientFlow)
- Right before sending request to target, right after getting response from target
- Optional: before and after Service Callouts
- Use Shared Flows and Flow Hooks for consistent use of Message Logging



Where to Add Message Logging

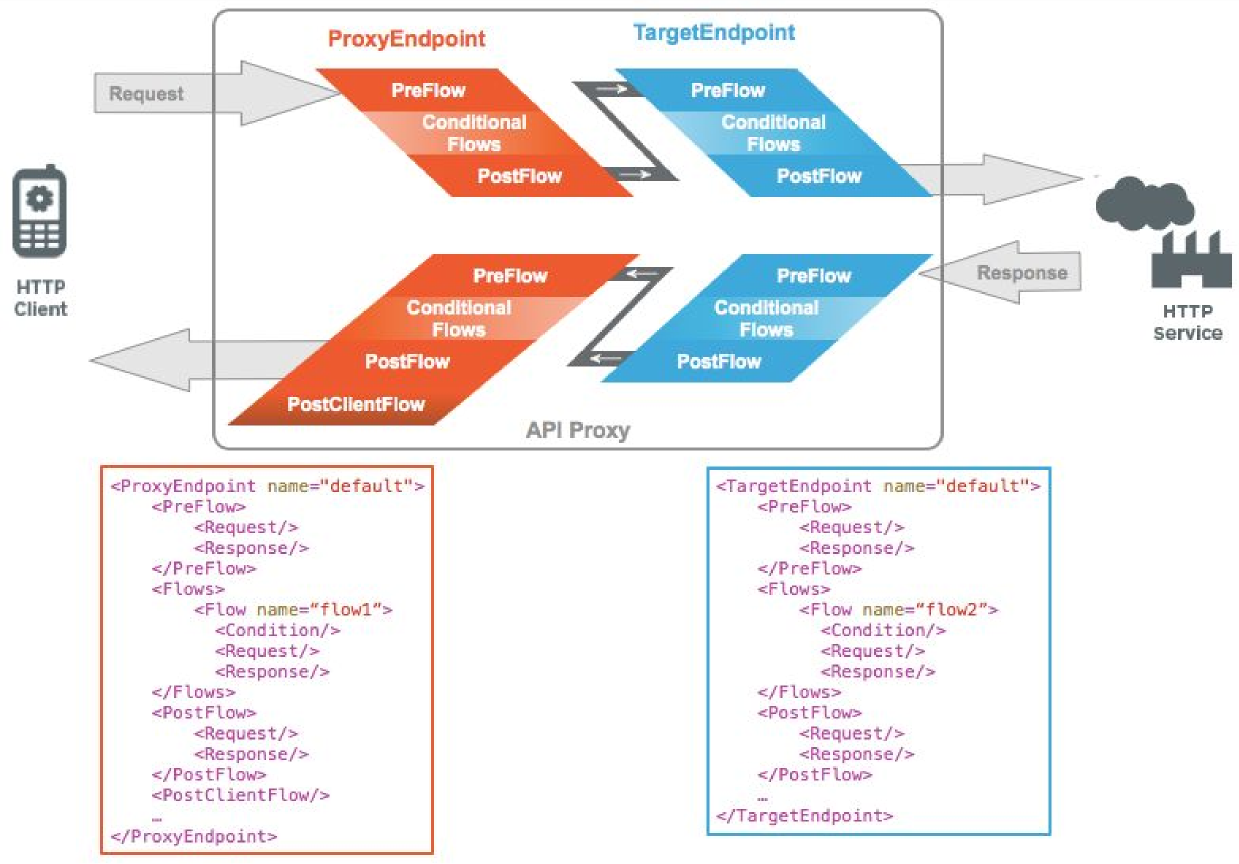
- Capture data at Flow Hook locations, log after response goes out (PostClientFlow)
- Capture data during Fault Handling and log in proxy default fault handling



Edge Logging Services

- Consider placing the policy in the ProxyEndpoint response, in a special flow called PostClientFlow.

- The PostClientFlow executes **after** the response is sent to the requesting client, which ensures that all metrics are available for logging and does not impact the response time to the client.



```
<ProxyEndpoint name="default">
    <PreFlow>
        <Request/>
        <Response/>
    </PreFlow>
    <Flows>
        <Flow name="flow1">
          <Condition/>
          <Request/>
          <Response/>
    </Flows>
    <PostFlow>
        <Request/>
        <Response/>
    </PostFlow>
    <PostClientFlow/>
    …
</ProxyEndpoint>
```

```
<TargetEndpoint name="default">
    <PreFlow>
        <Request/>
        <Response/>
    </PreFlow>
    <Flows>
        <Flow name="flow2">
          <Condition/>
          <Request/>
          <Response/>
    </Flows>
    <PostFlow>
        <Request/>
        <Response/>
    </PostFlow>
    …
</TargetEndpoint>
```

Edge Logging Services

- The Message Logging policy lets you send syslog messages to third-party log management services
- Some of the log management service providers are:
  - Splunk
  - Sumo Logic
  - Loggly
  - Stackdriver
- You can find all the above integrations on Apigee community

Edge Logging Services

For error handling, the best practice is to trap the errorcode part of the error response

```
{
   "fault":{
      "detail":{
         "errorcode":"steps.messagelogging.StepDefinitionExecutionFailed"
      },
      "faultstring":"Execution failed"
   }
}
```

```
<FaultRule name="MessageLogging">
    <Step>
        <Name>ML-LogMessages</Name>
        <Condition>(fault.name Matches "StepDefinitionExecutionFailed")
</Condition>
    </Step>
    <Condition>(messagelogging.ML-LogMessages.failed = true) </Condition>
</FaultRule>
```