Google Cloud

# Edge Advanced Mediation

## Extensions

# Extend with Programming

# Extreme Flexibility with Extension policies

- When you need more flexibility than supported by the out-of-the box policies, Edge has the ability for a developer to use server-side JavaScript or Java or Python.

- External libraries can be included at the organization level or via proxy.  This is defined with <IncludeURL> defined in the policy.

- Execution time limits available in policy to avoid infinite loop or slow performing code.

# JavaScript Object Model

**Context**

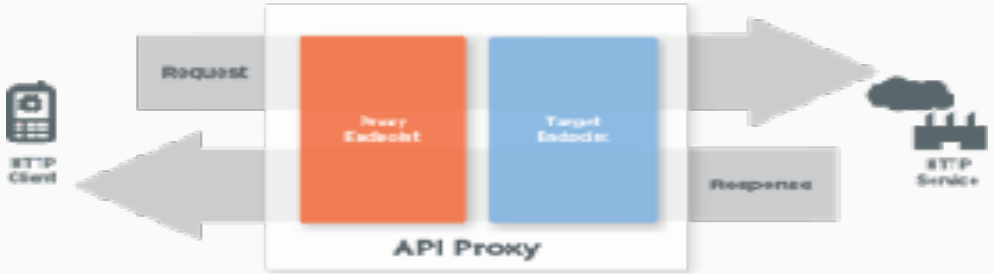
A context object (context) is created for each request/response.  Within the context, you can access variables and the HTTP message.

**Methods**

- getVariable(), setVariable() – *context.getVariable("myVar");*

**Messages Objects**

- request, response
- proxyRequest, proxyResponse, targetRequest, targetResponse



# JavaScript Overview

```
try{
    var policyCacheHit = context.getVariable("cachehit");
    var cachehit = "false";
    if (policyCacheHit == 1){
        cachehit = "true";
    }
    context.setVariable("response_cachehit",cachehit);
}catch(e){
  throw 'Error in JavaScript:' + e.toString();
}
```

Allows you to run server-side JavaScript to extend the capability of proxy processing. Important when needing to use loops/switches/complex logic.

- Preferred choice of callout amongst developers
- Can be used to leverage asynchronous httpclient requests
- Relies on Rhino
- Edge uses E4X, extending capability for XML support

# JavaScript Object Model (cont’d)

**Message Object Properties**

The HTTP message objects contain properties for each part of the HTTP Message:

- headers
- queryParameters
- method
- body/content

# JavaScript httpClient

The JavaScript http client can be used to make asynchronous http requests.

**Two methods exposed:**

- get()
- send()

**Each method returns an exchange object that exposes additional methods:**

- isError()
- isSuccess()
- isComplete()
- waitForComplete()
- getResponse()
- getError()

# Java Callout

Same principle as JavaScript, but using the java programming language.  Java is typically used when needing extreme performance with complex logic.

- Not available in free Edge organizations.

**Relies on two libraries:**

- expressions-1.0.0.jar
- message-flow-1.0.0.jar

https://github.com/apigee/api-platform-samples/tree/master/doc-samples/java-cookbook

*Network I/O, file system read/writes, current user info, process list, and CPU/memory utilization are not permitted by the security model.*

# Java Callout (cont'd)

```java
package com.sample;

import com.apigee.flow.execution.ExecutionContext;
import com.apigee.flow.execution.ExecutionResult;
import com.apigee.flow.execution.IOIntensive;
import com.apigee.flow.execution.spi.Execution;
import com.apigee.flow.message.MessageContext;

@IOIntensive
public class helloworld implements Execution{
    public ExecutionResult execute(MessageContext messageContext,
            ExecutionContext executionContext){

        messageContext.getMessage().setHeader("Content-Type", "text/plain");
        messageContext.getMessage().setContent("Hello World!");

        return ExecutionResult.SUCCESS;
    }
}
```

- Be sure to import the necessary Edge libraries. When compiling the jar for upload into Edge, however, do *not* include these as they already exist within the platform.

# Python Script

The Python Script policy lets you add customized Python functionality to your API proxy flow, especially when the functionality you need is beyond what the Edge out-of-the-box policies provide

- Not available in free Edge organizations and available in Edge Enterprise plan only

*Network I/O, file system read/writes, current user info, process list, and CPU/memory utilization are not permitted by the security model.*

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<Script name="Python-1">
  <DisplayName>Python-1</DisplayName>
  <ResourceURL>py://myscript.py</
ResourceURL>
</Script>
```

```
import base64
username =
flow.getVariable("request.formparam.client_id")
pwd=
flow.getVariable("request.formparam.client_secret")
base64string = base64.encodestring('%s:%s' % (username,
pwd))[:-1]
authorization = "Basic "+base64string
flow.setVariable("authorizationParam",authorization)
```

# Choosing Which Extension Policy to Use…

**Use built-in policies first and foremost (when possible)**

**Use JavaScript:**

- mashing up responses/manipulating json and non-complex XML
- looping /switching through datasets
- if it's more intuitive than Edge policies (for example, when setting target.url for many different URI routing combinations)
- Most preferred option for Edge developers

**Use Java/Python:**

- if performance is the highest priority
- when the solution requires functionality that is best served in Java (e.g. email notification service)

# Lab

**JavaScript**
**Java**
**Python**

# THANK YOU