AppSentinels.ai
Application Security Re-invented

# APPSENTINELS API SECURITY

## APIGEE DEPLOYMENT

## Revision History

| Version | Date | Revision Description |
|---|---|---|
| 1.0 | 22/08/2022 | The first version of the guide |
| 1.1 | 14/19/2022 | Service callout only based post client implementation |
| 1.2 | 25/10/2022 | Review comment incorporation |

## Scope

This guide contains approach and steps to deploy APIGee with Appsentinels controller.

## AppSentinels.ai Doc Center

The content in this guide is an excerpt from the topics in the AppSentinels.ai Doc Center. Visit the Doc Center for full context and more topics relevant to the AppSentinels API Security solution.

# Table of Contents

## Contents

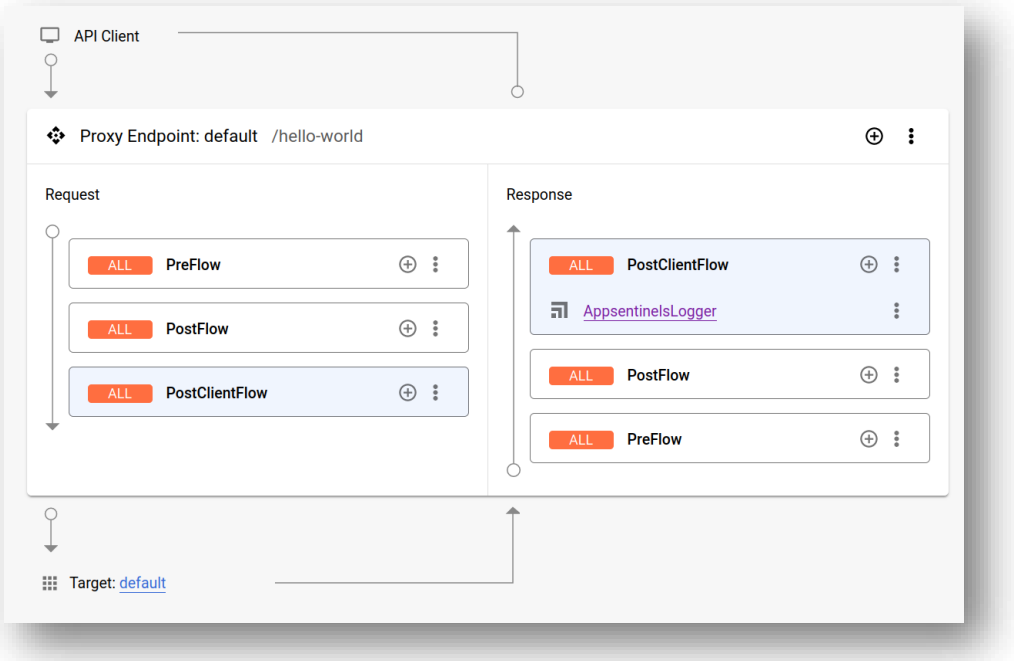
# Appsentinels logging requirements

Appsentinels will employ a controller on your premise to collect below information from APIGee,

1. Headers (name + values)
2. Query (name + values)
3. Path
4. Payloads of request and response

# Approach

In PostClientFlow, service callouts for request and response will independently pass on the headers along with other information (path, query, payloads) to Appsentinels controller endpoint. We will be using 2 Service Callouts since we cannot create one single request containing request and response information.



## Shared Flows:

1. Shared flows allow for easier integration into flows belonging to multiple API proxies
2. Service Callouts will be implemented as part of a shared flow which can be attached to any proxy's post client flow

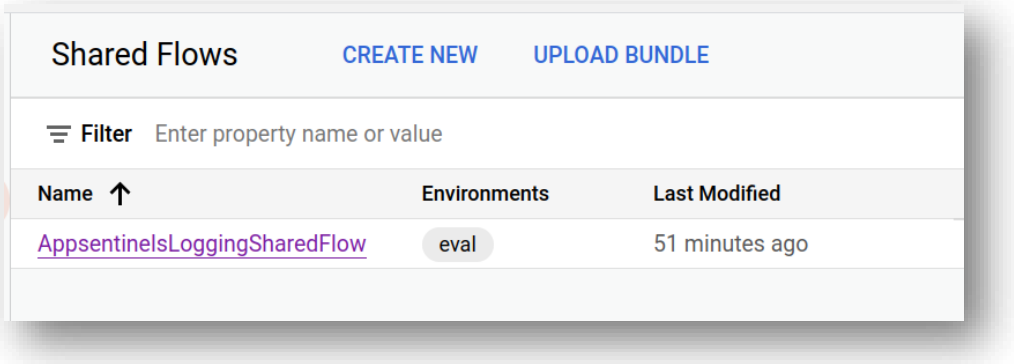
# Steps

## Creating KeyStore and TrustStore (in case of mTLS)

1. Create truststore **AppsentinelsLoggingTrustStore (only needed for self-signed certs in case server, aka Appsentinels controller is running a self signed certificate)**
2. Create keystore **AppsentinelsLoggingKeyStore** with alias **LoggingKeyStore** (https://docs.apigee.com/api-platform/system-administration/creating-keystores-and-truststore-cloud-using-edge-ui). Load client's private key and public cert into AppsentinelsLoggingKeyStore
3. These will be used for mTLS during Service Callouts

## Creating Shared Flows

1. There will be a shared flow bundles called AppsentinelsLoggingSharedFlow available (provided separately). This performs mTLS based request and response log sending to Appsentinels controller. This will be inserted into response PostClientFlow.
2. Now go to Apigee dashboard and Shared Flows. Next click on Upload Bundle
3. Select the AppsentinelsLoggingSharedFlow_*.zip bundle



4. Configure the endpoint for sending logs. Go to Service Callout-Request policy and configure the URL under HTTPTargetConnection
5. Go to Service Callout-Response and configure the URL under HTTPTargetConnection

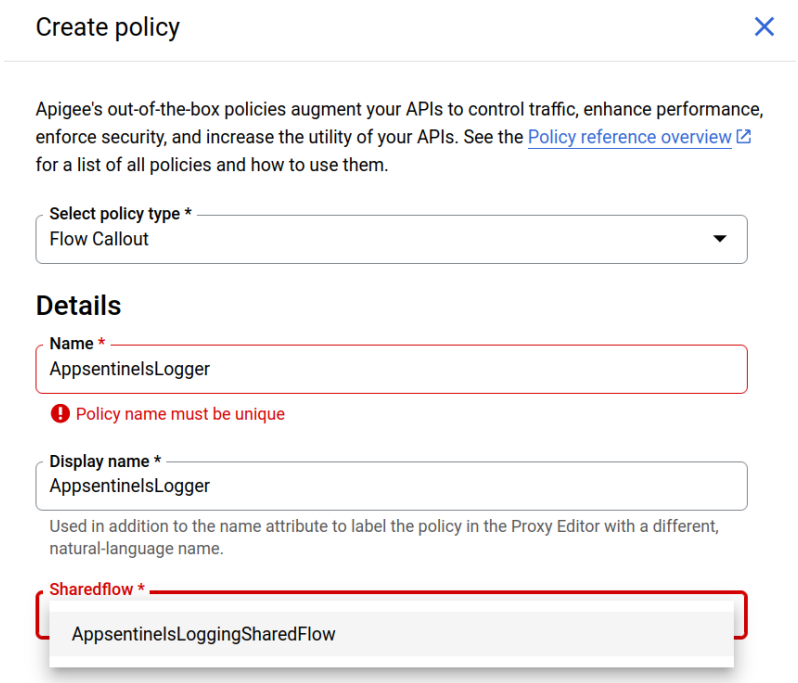
The below example configures URL to where Appsentinels controller will be reachable. This will be deployment specific.

```
<HTTPTargetConnection>
  <Properties/>
  <URL>https://<Controller-URL-here>:9006</URL>
  <SSLInfo>
    <Enabled>true</Enabled>
    <ClientAuthEnabled>true</ClientAuthEnabled>
    <KeyStore>AppsentinelsLoggingKeyStore</KeyStore>
    <KeyAlias>LoggingKeyStore</KeyAlias>
    <IgnoreValidationErrors>false</IgnoreValidationErrors>
    <Ciphers/>
    <Protocols/>
  </SSLInfo>
</HTTPTargetConnection>
</ServiceCallout>
```

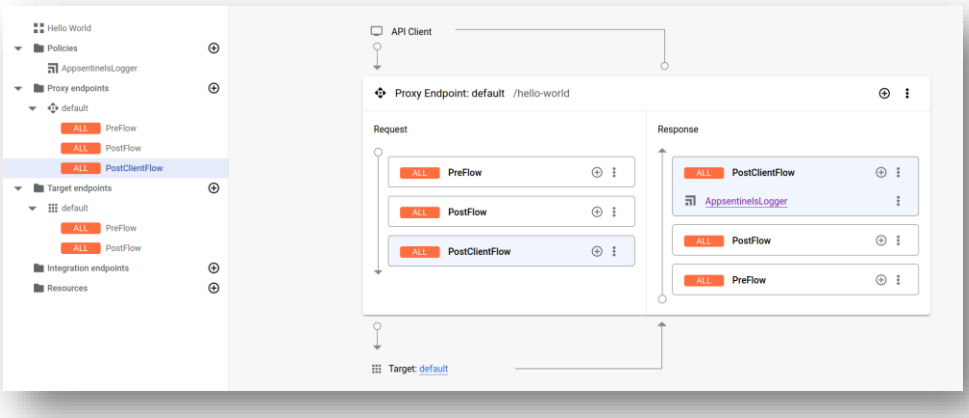
6. Now deploy the shared flows

## Attach Shared Flows

1. Go to your API proxy. We will be creating Flow Callouts and attach the above shared flows under Proxy Endpoints
2. In Response PostClientFlow, create a new policy of type Flow Callout. Rename it as AppsentinelsLogger. Under SharedFlow choose “AppsentinelsLoggingSharedFlow”. Add this policy unconditionally




3. The proxy flow should end up looking as below,




4. Deploy your API proxy with the changes