API Service Levels and OpenAPI

go/openapi-service-levels

Author: isaach@
Date: January 2019
Status: draft

Background

There is an active community initiative ("SLA4OAI"), driven by the Applied Software Engineering research group at the University of Seville, to "promote an open specification for service level agreement over REST APIs" in the form of an extension to OpenAPI.

The current proposal mixes a variety of separate API concerns — service levels, entitlements, and pricing.

This document is an attempt to define these areas, describe how Apigee approaches them, and lay out Apigee's position on extending OpenAPI in this area.

Definitions

API Service Level

At Google we use the following terms of art:

- **Service Level Indicator (SLI)**: a metric which can be used to assess one aspect of a system's reliability or performance. Availability is a service level indicator. So is 99th-percentile (p99) latency. An SLI is typically a time series, and may involve some level of sophistication (e.g., sliding windows) in its calculation.
- **Service Level Objective (SLO)**: a precise numerical target (often a ratio) for one or more SLIs, describing the minimum acceptable reliability or performance of a system. A given system may have different SLOs for different users, e.g., an internal objective and an external one.
- **Service Level Agreement (SLA)**: a contract with a user of a system or systems, usually including some combination of
    - a specification of nominal system behavior

    - SLOs for the system(s)
    - support levels for the system(s), e.g., response and resolution times for issues, typically tiered by severity
    - remedies available to the user in the case of missed targets, e.g., service credits, or a contract termination right

Notably, **SLIs and SLOs are technical constructs** whereas **SLAs are business constructs**.

API Entitlement

Outside access itself to an API, our customers specify API entitlement in two forms, which really differ only in the time period considered:

- **Quota**: an entitlement to API usage over a (usually relatively long) time period. For instance, a given API Product may offer a quota of 100,000 calls per month.
- **Rate Limit**: an entitlement to API usage over a (usually short) time period. For instance, a given API Product may allow no more than 10 calls per second per consumer.

API Monetization

Monetization of an API typically involves:

- **Metering**: recording API usage in sufficient detail to perform rating.
- **Rating**: converting records of API usage into an owed amount of money. This conversion may involve simply a fixed charge per API call, or considerably more complex schemes (see below).
- **Billing**: presenting to an API user a report of amounts owed, taking into account any discounts, service credits, taxes, and revenue sharing.
- **Collection**: receiving and recording payments of amounts owed by users of APIs.
- **Enforcement**: preventing a user from using an API once they have exhausted their pre-paid service credit, or reached a credit limit.

Apigee supports all of the above, and in particular has **very flexible rating** support, including provision for

- fixed fee per API call
- fixed fee per time period

- volume-based tiers of fees per API call
- volume-based bundles of API calls
- revenue sharing schemes
- charging variable amounts based on arbitrary runtime attributes:
    - parameters in the request
    - elements of the response
    - time of day, day of week
    - source geography
    - current load on the API
    - etc.

We find many exotic combinations of the above rating schemes in the wild of our customer base.

Thoughts on extending OpenAPI

At Apigee our current priority would be to codify **SLIs and SLOs** for APIs in a formal description language extending OpenAPI. Based on such a codification our tooling could then offer richer native support for the user stories below.

We believe that **SLAs**, as well as not being readily amenable to such a codification, probably don't belong in OpenAPI in any case.

A codification of **entitlements** in an OpenAPI extension would definitely be interesting to us, but isn't a pressing priority.

We'd suggest that **monetization** definition be a separate initiative. In the real world there is significant complexity in rating API usage, likely deserving of its own OpenAPI extension.

Some example user stories

As an **API developer providing an API** I would like to:

- **better understand what SLOs I can reasonably target**
  so that I can **offer an SLO for my API**

- **better understand the performance of my downstream dependencies (e.g., back-ends)**
  so that I can **determine their effect on my SLO**
- **better understand the performance of policies in my proxy**
  so that I can **determine their effect on my SLO**

As an **API product manager** I would like to:

- **know the SLOs of my downstream dependencies**
  so that I can **create products which meet my customers' needs**

As an **API operator** I would like to:

- **have alerts automatically set based on SLOs, to alert me of risk of missing the objective**,
  so that I can **take remedial action**
- **receive regular reports detailing API performance against SLO**,
  so that I can **report to the business owners**
- **easily see both the internal and external SLO commitments for various APIs or Products**
  so that I can **quickly categorize and prioritize** my operational efforts

As an **API consumer** I would like to:

- **know what service level is offered**
  so that I can **make an informed decision about adopting the API**
- **understand historical actual performance of an API**
  so that I can **understand how reliable I might expect them to be**
- **know that I'm getting the service level I deserve, and/or am paying for**,
  so that I can **claim remedies if SLOs are not met**