API Versioning Strategy

Google Cloud

Versioning is needed when...

- change would break existing software
- new fields are required in payload
- previously available data is removed

apigee

Some versioning approaches

Date parameter used to look up the version of an API that was valid at that time;

GET /**2017-04-23**/accounts

apigee

Some versioning approaches

Use a custom header;

GET /1/accounts
x-api-version: **1.1**

apigee

Some versioning approaches

Version indicator (v20) somewhere in the URI;

GET /**v20**/accounts

Version Approach Pros & Cons

Approach	Pro	Con
"Date" based filter; GET /**2010-04-01**/accounts	Resolves to whatever version is active	Unfamiliar to most developers. May be hard to implement.
Version in header; GET /1/accounts x-api-version: **1.1**	Semantically beautiful. Adheres best to Fielding's REST ideals.	Not as easy to explore with common tools.
Version in URI; GET /**v20**/accounts	Clear, familiar to developers. Easy to work with in browsers and tools.	May be confused for a version of the resource, not an API version.

apigee

Preferred Approach

Version in URI;
GET /v1/<domain>/<resource>/{id}

or

GET /<domain>/v1/<resource>/{id}

- **Simple**
- **Unambiguous**
- **Intuitive**

apigee

Versioning Communication

- Proactive prior to release
- Leverage API Portal
- Provide details and context
- Provide deprecation details for older versions, if applicable

apigee

Be Consistent

Consistency leads to **predictability**

Predictability leads to **intuitive usability**

Usability leads to **usage**

Usage leads to **innovation**

apigee

Thank You

Google Cloud