Taming Your API
Sachin Agarwal, Principal Product Manager
How are we writing APIs? What kinds of APIs do we have?
Internal APIs

External APIs
What Makes A Bad API?
What Makes A Good API?
Example Code
In all the different languages under the rainbow.

Documentation
In all the different languages under the rainbow.

SDKs
In all the different languages under the rainbow.

Verbose Error Codes
If it’s wrong, guide me to how to do it right

Sandboxes
Try before you buy
Predictable
- Immutable IDs
- Guaranteed reproducibility
- Output format matches the input format

Available
- External APIs are third party dependencies
- Engineer your application to degrade gracefully

Performant
- Every external call is yet another roundtrip
- Where are the API calls being served from? Your AZ? A CDN?
Good APIs

Bad APIs
Good APIs Have All Of These

- Great Documentation
  - Examples in Multiple Languages
  - Verbose Error Codes
  - Explicit Responses

- Predictable
  - Immutable IDs
  - Consistent Responses
  - Nouns for Endpoints
  - Consistent Casing

- Available
  - Five Nines+ Uptime

- Flexibility
  - Pagination
  - Filters
  - Sort Orders

- Performant
  - Quick for the User
  - Quick for your App

Consistent Responses
Immutable IDs
Nouns for Endpoints
Consistent Casing
Predictable
Available
Performant
Flexibility
Great Documentation
Knowing This, Why Do We Make Internal APIs That *We Know* Are Bad APIs?
Internal APIs Rarely Have Any Of These

Great Documentation
- Examples in Multiple Languages
- Verbose Error Codes
- Explicit Responses

Flexibility
- Pagination
- Filters
- Sort Orders

Predictable
- Immutable IDs
- Consistent Responses
- Nouns for Endpoints

Available
- Five Nines+ Uptime

Performant
- Quick for the User
- Quick for your App

Consistent Call and Response Shapes
Well-Defined Rate Limits
Be Excellent To Your Team And Your Future Self
How Can We Make This Internal API Better?
Who Uses the API Internally?
Everyone on the engineering team!

Who Will Use the API Externally?
Some folks!

How Are Those Folks Different?
The folks outside the building can’t Slack the other folks to figure something nonobvious out
Taking A Private API Public

Analyze First

Getting Prepared

What Would External Users Expect?
Endpoints, formats, plural/singular, surfaced in SDKs

Incremental Delivery
Which endpoints do you do first? Which can come later?
Land and push them as you build them

What Analytics Should You Build?
Mixpanel? Honeycomb? New Relic? Loggly?
Write The Docs First

I know, it’s not fun. But, seriously, do it first. It makes it a lot easier to validate that this new API looks like your other ones.

Calls *And* Responses

Great API docs have example code and example responses. Your sample code shouldn’t be from your actual internal app.

Verbose Error Codes

Use more than 400s and 500s
Have human-readable descriptions with suggestions for the most likely way people will malform requests
Taking A Private API Public

Analyze First

Getting Prepared

Write Your Documentation

Progressive Delivery

Mix And Match
If you're making private APIs public, you'll have to change your internal application code too - don’t forget and don’t clone!

Friendly Strangers
Let folks try things out at hackathons
Last chance to change your docs and shapes

Broad Release
Public docs, SDKs, example snippets all need to go live
Flip one flag and release them all simultaneous (obviously)
Monitor
Check usage and make sure that you haven’t accidentally introduced anything too expensive to service. (Progressive Delivery is your friend here as well.)

Update Your SDKs
You’re inevitably going to extend a successful public API. Don’t forget to update your SDKs too!

Accountability
Revisit use of your APIs every month or every quarter. The way people use your APIs when you release is different than the way people use them after they’ve baked a while.
There Should Be No Difference In How You Build
Many private APIs become public eventually. It’s a best practice to write the docs first, make calls and responses consistent, and be a good citizen from the beginning.

Transitions Happen
You can force your co-workers to upgrade your internal APIs, but you can’t force customers to upgrade your external ones. Be smart!

Roll Them Out In Stages
Progressive and incremental delivery - ring deployments and shipping smaller things - are your friends. Feature flags can help.
Thank You!