REBOOTING API’S AT SCALE

JAYADEBA JENA
HEAD OF API DESIGN @ PAYPAL

RAHUL DIGHE
CHECKOUT API PRODUCT @ PAYPAL
PAYPAL'S API JOURNEY

2004
PayPal launches one of the 1st NVP/SOAP based APIs

2013
PPaaS (PayPal-as-a-Service) setup with a charter to provide governance and accelerate migration to REST APIs. First version (v1) launched.

2018
Investment in v2 of our APIs begins. First set of v2 APIs to launch soon.
SIGNs THAT YOUR API MIGHT NEED A REBOOT

- You have more features/capabilities but your competitor steals the show because their API is easy to understand and integrate with.

- API were built for a specific market/vertical use case and now other partners are asking for it but it’s not reusable.

- API’s have become bloated with fields/objects that have organically grown and no one quite remembers why you have it in the first place.

- API’s were designed for a country/locale in mind but then the business grew internationally.

- API interface is not a business abstraction (instead matches backend system implementation).

- API behavior is prone to idiosyncrasies of your underlying system.

- Support engineers are critical to the success of your integrations.

“IT’S TIME TO THINK OF API’S BEYOND JUST AN INPUT /OUTPUT MECHANISM TO SOLVE AN IMMEDIATE INTEGRATION NEED.”
SKEPTICS IN THE INDUSTRY

THE REBOOT SKEPTICS

- It works – if it ain’t broke don’t fix it!
- Priority is to launch a new mobile app, and optimize our website
- We have the best capabilities, sales engineers can bridge the gap if a partner needs help
- Should you not fix the bugs that you have instead?
- Our competitors are not doing it.
- No one will migrate, are we now going to maintain 2 sets of APIs?
- We have never deprecated an API, what signal would it send to our existing customers
- Feedback is positive (from the business person) I showed our API’s too, it does everything they want it to do
- Why did we mess up in the first place?
API PRODUCT PROCESS

**DISCOVERY**
- Problem Identification
- Competitive Research
- Developer Experience
- Build Persona
- User Research
- Business Case Development

**DESIGN**
- Define API endpoints
- Fields / Objects
- Documentation
- Validation errors
- API Specification (e.g. swagger)
- Mock Request Response

**DEVELOP**
- Define Backlog
- Generate code from Api spec
- Developer Experience
- Security
- Load & Performance
- Functional Tests
- Iteration

**DEPLOY & LAUNCH**
- SDK Generation
- User Testing
- Monitoring
- Launch
- Integration Guides
- Marketing collateral
- Training
- Webhooks
- Sandbox
SEEK OUT YOUR CRITICS

What's your take on the v2/api design?

!@#$!
DON’T LOOK BACK UNLESS YOU WANT TO GO THAT WAY
API FRAMEWORK/GUIDELINE

- Setup a framework / guidelines encompassing the product life cycle:
  - Design
  - Develop
  - Document
  - Launch/Release
  - Manage lifecycle, of your APIs

- If you do not have the resources to define your own be inspired by others [https://github.com/paypal/api-standards]

- Such approach provides guiding principles and solutions, taking the mystery out of building API products and make them more accessible.
API VERSIONING

VERSIONING POLICY

- Do you have a clear versioning policy:
  - that describes the product evolution
  - compatibility guidelines/principles
  - EOL (end-of-life) policy
- Versioning guidelines for your internal developers are exhaustive than the public guidelines (do your developers understand it?)

<table>
<thead>
<tr>
<th>Versioning</th>
<th>Major version</th>
<th>Minor version</th>
</tr>
</thead>
<tbody>
<tr>
<td>External (v1)</td>
<td>{1}</td>
<td>{1}</td>
</tr>
<tr>
<td></td>
<td>{1}</td>
<td>{2}</td>
</tr>
<tr>
<td></td>
<td>{1}</td>
<td>{3}</td>
</tr>
<tr>
<td></td>
<td>{1}</td>
<td>{4}</td>
</tr>
<tr>
<td></td>
<td>{1}</td>
<td>{5}</td>
</tr>
</tbody>
</table>

- Major version
- Minor version
Every company should define the core principles of an API platform.

API standards are written around the platform principles and provide guidance to all stakeholders.

API standards define all patterns, style guide, helps maintain consistency, helps adhering to company’s security policies, versioning, backward compatibility, lifecycle management etc.
TOOLS & PROCESSES

- Educate & Train on API Design
- Common objects/capability model
- Centralized API Repository
- Contract vs implementation verification
- API Maturity Model
- Dev Frameworks Spec to Code generation tools
TRY NOT TO SURF IN CALM WATERS

TIMING IS EVERYTHING

- Did you just take on a new role and people are more receptive to hearing new ideas from you?
- Is there a new leader that has joined the organization that has a passion for APIs?
- Did one of your competitors just launch an API that was well received by the market?
- Company is in a growth mode and is open to investment for a Proof of Concept.
DOES YOUR API NEED A REBOOT?

REACH OUT WE ❤️ APIS & ARE HAPPY TO HELP

jjena@paypal.com
rdighe@paypal.com
@rahuldighe