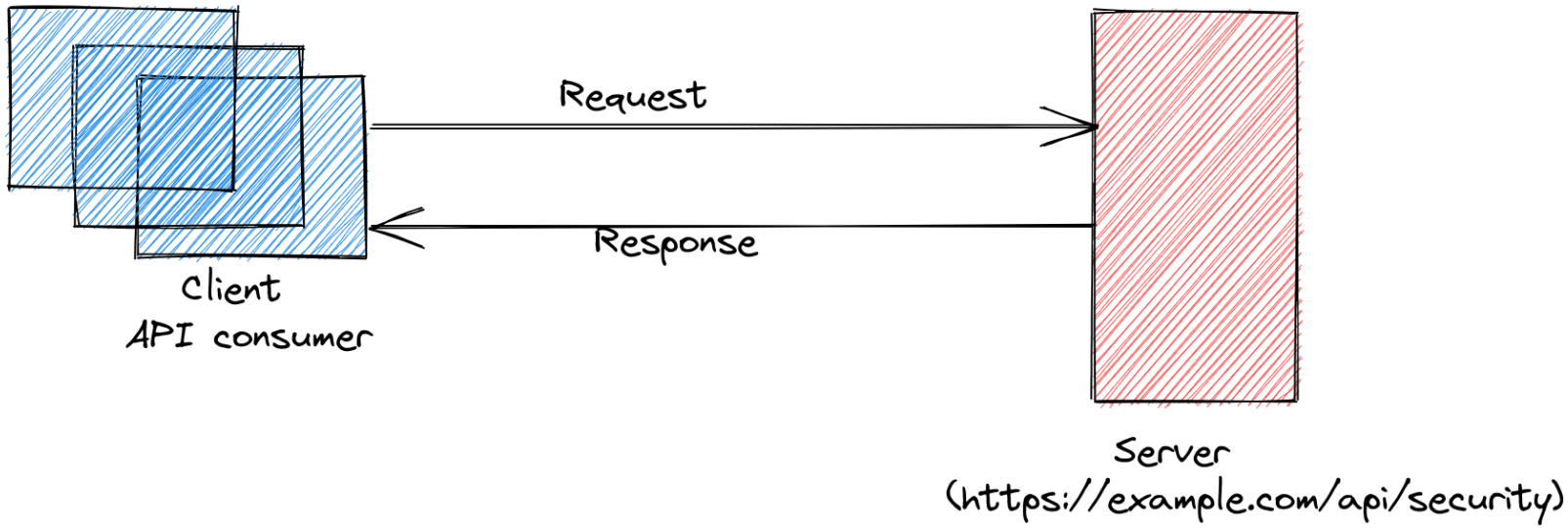
How (Not) To document Your APIs

In simple terms, What is an API?

- *An interface through which clients interact with the services (server).*



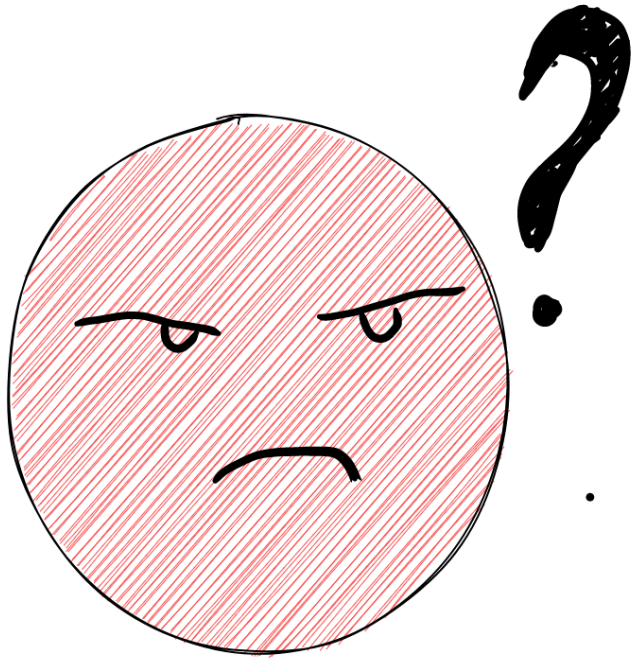
Popular tools in 2015

Swagger

API Blueprint

Popular/Emerging tools in 2022

Why is it still hard to understand API, even when we have lot of good tools?



Instance 1: Account Details API

- This API is used to get account details based on customer ID (Required) and account id (Optional).
- API documentation says the customer ID field is mandatory, but if not provided still works.
- Secondly, it also returns other resources which have matching account IDs but different customer IDs. It was not documented for the API.
- Even if it returns details for multiple resources, there is no way to identify which resource belongs to which customer ID as there is no identifier field in returned resource details.

What could have been improved here?

- Make it clear in API documentation for possible scenarios for customer ID/account ID.
- If it is not required, then don't used mandatory term.
- If you are going to return multiple account details for multiple customer for customer which is ok, then provide a way to find the requested customer details in the returned response like using customer ID.
- Try to have a documentation review process or automate this things.

Instance 2: Suspension Details API

- Legacy API is used to get the **suspension details** for customer id.
- API is like GET BASE_URL/suspension-details, but it returns account details.
- Due to bad naming, it created more confusion during migration. It should have been named GET "BASE_URL/**account-details**"

What could have been improved here?

- It is better to name the API for what it does not for what it will be used for.
- Let the consumer of the API decide how it want to used that.

Instance 3: Money Transfer API

- An API used to **send money** from an internal account to an external account (customer account).
- API has some fields marked as mandatory in the documentation, but it can accepts empty/null values.
- Details for **debtor** and **creditor accounts** have all the fields marked as optional. In reality, as per the business, that API needs mandatory details, but technical implementation says different things.
- In the name of adding encryption features in the API, many fields are renamed, new fields were added, and some were removed.

What could have been improved here?

- API which involve money, should be thoroughly reviewed by multiple folks.
- It should mark fields that is needed mandatory not keeping optional.
- Provide those details strictly in documentation not blindly copy paste.
- If you are going to update contract of the API while adding encryption, which is fine, better update the documentation as well as acceptable value in the documentation and communicate the same to other stakeholders.

Instance 4: 2 Factor Authentication (2FA) API

- An API is used to send OTP messages to the customers, and another API is used to validate OTP messages. Validating OTP requires a sent OTP pin and function ID(*used to identify message format specific to the product*).
- After 8 months from development of that code, frontend folks pushed small change which was for product code update, and then smoke started failing. We were confused, why this broken the OTP validation API?
- Later, after debugging and some calls with other team, we realized that function ID need to be same which was previously taking default value (**was not sent in the request**).
- The function Id is required and need to be the same in OTP generation, and validation calls for a product code. This was not documented in tools like Jira, confluence, etc.

What could have been improved here?

- If during development of the feature in the product, you found that documentation missed important details which could be easily missed by the other developer, document it somewhere.
- Function ID is mandatory so better not accept the request with missing function ID. It will make the life easier, and developer don't have to check the API code to understand the actual cause.

Is there any better way to manage this?

- I have used postman, which allows me share API details and add test code to verify the response.
- I can easily share the collection file to team or commit in git repository.
- I recently started using swagger UI, and it really solves the core problem of managing and updating the API documentation.
- It automatically creates basic documentation template.
- It will only document what is implemented in the API, which seems fair, but again it is the developer who need to make sure he is implementing the contract properly then only this tools will provide meaningful details.

Example for Spring-boot: Swagger UI

- **Dependency**:

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.11</version>
</dependency>
```

- **URL**: http://server:port/context-path/swagger-ui.html

security-controller

POST `/security`

Parameters | Try it out

No parameters

Request body required | application/json

Example Value | **Schema**

SecurityRequest {

isin*	string
schemeName*	string
amfiCode	string
comment	string

}

Responses

Code	Description	Links
200	OK Media type: */* Controls Accept header. **Example Value** | Schema	*No links*

```
{
  "securityId": 0
}
```

Thank you

Pravin Tripathi
Application Developer

Connect:
Blog: https://pravin.dev/
Twitter: @pravin_yo
Medium: @pravinyo
LinkedIn: /in/pravin-r-tripathi