**API Best Practices**

# Definition

## Swagger Definition

* When naming a reference tag ($ref) use a prefix like “ref-“ to avoid confusion

Some of the best practices we use/recommend in API development are:

* API Definition: We use industry standard specifications for API definition (e.g. OAS, RAML). We treat API definition as a development artifact and ensure version control around it. We extensively use external references or inheritance to manage reusable pieces of the definition.
* We also recommend and conduct design reviews for APIs being developed to ensure they align to best practices around API security, design, error handling and SLA management
* API Scope: APIs should be defined around business capabilities with clearly defined scopes to align to microservices architecture principles
* API Versioning: We recommend API version be specified in the API URL. API version change should indicate a change in interface and not simply a change in implementation
* API Design: We incorporate some of the best practices in API design, which can make it easier to consume. Some such examples are design consistency, support for multiple payload formats, defining data retrieval scopes and filters, pagination, etc.
* API Error Management: While exception handling is important from an application architecture /development perspective, we feel for APIs it is equally important to be able to communicate the error back to the caller in a meaningful way that can help troubleshoot. We use a combination of http error codes, exception logging, error blocks and concept of ‘soft’ & ‘hard’ errors to handle and communicate errors gracefully
* API SLA Management: Based on the use-case, we have used distributed caching at various layers to improve on API performance. We also recommend use of alternate paths to manage API availability
* API Security: Mask sensitive data if possible. We feel TLS is good enough to secure data in transit in most cases. Encryption, if absolutely necessary, should be done at the data level. APIs should be secured through standard security mechanisms like mutual SSL or OAuth, rather than custom security models. API management tools should be leveraged for threat protection