For your API style guide to be most effective, it should cover several common API design themes and be written in a way that is:

- Clear
- Easily Updated
- Actionable
- Enforceable

According to an excellent article by Nordic APIs on the best practices for creating an API style guide, here are some of the common design themes you should include in your API style guidelines.

Design Style and Specification

One of the first points to cover in your API guidelines is defining the overarching architectural style. Should your organization or team follow REST, GraphQL, or a different design pattern?

In addition, you'll want to let developers know whether they should be describing their APIs with some kind of specification (hint: they probably should), and if so, what that specification is.

Authentication and Authorization

One important aspect of API design is security. In addition to design style, security should be one of the major themes in your style guide. In particular, style guides should explain how developers should implement authentication and authorization.

Some of the top authentication frameworks include:

- Basic Auth
- OAuth v1
- OAuth v2
- OpenID
- SAML
- TLS
- JSON Web Token (JWT)

Usually, OAuth v2 is the industry-standard protocol for authentication.

Path naming is important although some types of API (especially REST/Hypermedia APIs) consider an URL to be opaque, developers exploring an API will often type those paths whilst playing around with an API, and having consistent naming conventions and guessable URLs improves integration time. Make your URLs simple, consistent, and universal.

Error Codes

Consider HTTP error codes as another critical component of your API guidelines. For example, some teams may decide to use `400 Bad Request` for data validation errors, others may use `422 Unprocessable Entity`, and maybe one team reads the HTTP spec wrong and decides to incorrectly use `406 Not Acceptable`. Your style guide can pick one, and generally help people make better quality API errors to help computers and humans.

Versioning

Versioning and breaking changes and how to treat them can vary between organizations. Be sure to highlight how to treat versions and breaking changes in your API guidelines, and update this section as your versions evolve.

Units, Formats, and Standards

Define the units, formats, and standards that developers should follow in your API style guide. What to define can depend on your industry, but “some types of data — like date-times — are relatively universal.”

[Source: 11 Tips for Creating an API Style Guide]

API Style Guide Best Practices

Now that you know what to include in your API guidelines, here are some best practices for creating excellent style guides that your organization will adopt, contribute to, and want to use.

Link to External Resources

Create an easy-to-consume style guide that is brief yet thorough. Developers should be able read the entire style guide, but it should also be comprehensive enough to provide consistency for even small considerations.

One way to create a more comprehensive style guide is by, “frequently linking to external resources, where specific standards or best practices are described in more depth. These can be your organization’s own resources or even trusted third-party resources.”

Include Request and Response Examples

One of the best ways to illustrate and show an idea is to include an example. Build in sample requests and responses throughout your style guide. Sometimes it helps to include several requests and responses to show how developers should—and should not—design their APIs.

Encourage Collaboration

Work across your team to collaborate on your API style guide. Encourage developers to submit feedback or request help on their API designs. Cross-team collaboration will improve buy-in and increase adoption of your guidelines.

Source

https://stoplight.io/api-style-guides-guidelines-and-best-practices