API Anatomy

This repository is an attempt at defining a set of application level terminology for HTTP API design. Along with each term is a description and an OpenAPI description of an example API demonstrating its use.

The intent of this collection is not to suggest best practices but to be a set of scenario examples that tooling developers can use to verify that they provide broad support for API designers.

Resource Design

A HTTP API is a set of resources, identified by URL, that share a common base URL.

A HTTP API is a set of resources that a consumer interacts with to achieve some desired objective. Each resource is identified by a URI. That's why it is called a Universal Resource Identifier. All resources in a HTTP API share a common base URL. Most API designers these days use just the path portion of the URL to identify the domain concept of interest. The authority part, i.e. schema and host, are just used to identify where the service is deployed, and the query parameters are used to manipulate the response in some way to change how the domain concept is presented.

Path segments

Kind	Description
Category	Organizational segment used to group related resources together in the hierarchy. e.g. `https://example.org/accounting/invoices`
Data Partition	Segment used to partition different sets of similarly structured data. e.g. `https://example.org/acme/employees`
Version	Segment used to partition resource hierarchies due to breaking changes in resource structure or behavior. e.g. `https://example.org/v1/users`
Key/value pair	A segment pair used to attach some value to resources in the sub hierarchy. e.g. `https://example.org/subscription/XHJSD-JSDS-UERJE/resources`
Collection	A segment used to indicate a set of some domain concept. e.g. `https://example.org/movies`
Item	A segment used to indicate a specific instance of some domain concept within a set. e.g. `https://example.org/movies/some-movie-id`
Relationship	A segment used to indicate a relationship between one domain concept and another. e.g. `https://example.org/movies/some-movie-id/actors`

Kind	Description
Multi-segment	A set of segments used to allow API consumers to define a hierarchy for a set of domain concepts. e.g. `https://example.org/bookmarks/personal/cooking/testkitchen`
Matrix	A rarely used capability to insert a multi-dimensional table of domain concepts into a hierarchy. e.g. `https://example.org/countries;continent=asia/cities`
View	A path segment used to explicitly indicate an alternate resource for the same domain concept. e.g. `https://example.org/recipies/eggnog;printable`
Action	A segment used to perform an unsafe operation on either a related resource or some kind of data processing on the transfered information. `https://example.org/me/invites/accept`
Function	A segment used create results that are usually an aggregation of domain concepts or some derivative of a domain concept. `https://example.org/me/documents/recentlyUsed`

Behavior

The set of path segments defines a hierarchy. Path segments provide context to the following segments. This can have a significant impact for some types of path segment. A version segment at the far left of a path, has a very different impact than one at the far right of the path. The hierarchial nature of a path is the major differentiator between path segments and query parameters.

Query parameters

Query parameters are used to create variants of the resource identified by the path. Each of these variants is technically a resource on its own as it has a distinct URL. However, generally, variants created by query parameters will share many of the same characteristics of the resource identified by just the path.

Kind	Description
Projection	Identifies a resource that contains a subset of the information contained in the path-only resource. Consider this a vertical slice of the path-only resource. https://example.org/books?select=Title,Author
Filtering	Identifies a resource that contains a subset of the items from of the path-only resource. Consider this a horizontal slide of the path-only resource. e.g. http://example.org/books?author=Smith|

Kind	Description
Sorting	Identifies a resource that contains a set of items from the path-only resource in some specified order. e.g. `https://example.org/tasks?orderBy=Deadline`
Range	Identifies a resource that contains a subset of items from the path-only resource based on a ordinal constraints. e.g. `https://example.org/tasks?top=10&skip=50`
Transclusion	Enables a caller to include a representation of a related resource into the representation of the path-only resource. e.g. `https://example.org/users/22?$expand=manager`
Sparse identifiers	Identifies a resource using n property values within a n-space. e.g. `https://example.org/map/image?lat=100&long=50`
Behavior modifiers	Identifies a resource that is a variant of the path-only resource in some domain specific way. e.g. `https://example.org/me/calendar?view=week`
Representation format	Identifies a resource that is a variant of the path-only resource in the format of the reprentation. e.g. `https://example.org/me/tasks?format=CSV`

URLs are serialized as strings when being used by HTTP. However, the values used for parameters in the URL often have additional semantics that do not have standardized string serializations.

Parameter Type	Serialization Notes
strings	Serialization examples include: `apple` or `"apple"` or `'apple'`
arrays of strings	Some examples of ways that arrays of strings can be represented include: `"apple,banana,pear"` or `["apple","banana","pear"]` or `apple,banana,pear`
expressions	Examples in query parameters include `filter=fruit eq "apple"` and `filter[fruit]=apple`
json	
dynamic form data	
dates	ISO, unix
numbers	

Parameter Type	Serialization Notes
Constrained types	enums?

Behavior

Beyond the kind of parameters and data types of individual parameters, there are often dependencies between parameters that appear in a URL.

Type	Serialization Notes
Optional parameters	Optional path parameters can be challenging unless they are a key/value segment pair. Query parameters are commonly optional. Making query parameters required is a burden on client applications and can introduce challenges with relative references.
Dependent parameters	Due to the natural hierarchy of path segments, path parameters are frequently dependent on the value of parameters higher in the hierarchy. Interdependencies between query parameters are possible, but they are difficult to describe in metadata and hard to enforce in client code.
Parameter ordering	Most APIs do not enforce any ordering of query parameters. However, technically, different parameter orders are different URLs and therefore different resources. This mainly affects intermediaries like caches, that need to do some normalization to prevent multiple copies of representations.
Parameter name casing	Query parameter names are technically case sensitive, but in practice they rarely are. Intermediaries will often need to normalize URLs to determine a canonical URL.
Encoding issues	

Requests

Feature	Description
Prefer header	The prefer request header can be used for a variety of purposes including suggesting the size of response representation to be returned, or to indicate a preference for a long running operation.
Accept	An accept header can be used by a client to request different representations (i.e. different media types) of the same resource. This is often also achieved using a

Feature	Description
	query parameter.
Idempotency	Some HTTP interactions are by definition idepempotent. Others can be made idempotent by including a unique value in the request. This can be important for requests that must only execute once, even with an unreliable network.
Authenticated Interactions	Some resources should only be accessible by clients that have the appropriate authentication. Ideally, the `Authorization` HTTP header is used for this. There are some alterative solutions for certain circumstances.
Paid interactions	API providers sometimes choose to charge for a client to make an API request. APIs should indicate when a payment mechanism must be provided. Identifiers of payment mechanisms can be provided in the request, or can be tied to an authenticated identity.
Detecting changes in data	API consumers often want to retrieve changes to a resource, since some prior point in time. There are a variety of ways this can be achieved.
Tunneled requests	This is sometimes an anti-pattern where part of the request message, either headers or body, is used to identity what could be identified in the URL. There may be valid reasons for doing this but it is important to understand the trade-offs of not identifying the distinct resources via the URL.
Batches	Batching is an application level solution to being able to make multiple requests in a single HTTP message. This can be a performance optimization for HTTP/1.1 based requests due to the limited number of TCP/IP connections allowed from some clients. However, it is rarely the right solution when using HTTP/2.

Responses

Feature	Description
Redirects	
Throttling	
Long running	
NoContent or Content	Some HTTP interactions return a representation of the resource that is being interacted with, others do not. Whether content is returned or not is generally based on a combination of the method and the status code.

Feature	Description
Partial failure	
Redirects	
Cachable content	
Deleted	

Representation Design

Feature	Description
Media types	
Multi format responses	
Batch	
Polymorphic responses	
Null vs absence	
Deprecation	
Collections vs items	
Facets	
Annotations	
Links	

Other

- Security
- Privacy
- Workflow (sequences of requests)
- Errors