

---

# REST API Design Rulebook

*Mark Massé*

---

# Table of Contents

<b>Preface</b> .....	<b>ix</b>
<b>1. Introduction</b> .....	<b>1</b>
Hello World Wide Web	1
Web Architecture	2
Client–Server	3
Uniform Interface	3
Layered System	4
Cache	4
Stateless	4
Code-On-Demand	4
Web Standards	5
REST	5
REST APIs	5
REST API Design	6
Rules	6
WRML	7
Recap	7
<b>2. Identifier Design with URIs</b> .....	<b>11</b>
URIs	11
URI Format	11
Rule: Forward slash separator (/) must be used to indicate a hierarchical relationship	12
Rule: A trailing forward slash (/) should not be included in URIs	12
Rule: Hyphens (-) should be used to improve the readability of URIs	12
Rule: Underscores (_) should not be used in URIs	12
Rule: Lowercase letters should be preferred in URI paths	13
Rule: File extensions should not be included in URIs	13
URI Authority Design	14
Rule: Consistent subdomain names should be used for your APIs	14

Rule: Consistent subdomain names should be used for your client developer portal	14
Resource Modeling	14
Resource Archetypes	15
Document	15
Collection	15
Store	16
Controller	16
URI Path Design	16
Rule: A singular noun should be used for document names	17
Rule: A plural noun should be used for collection names	17
Rule: A plural noun should be used for store names	17
Rule: A verb or verb phrase should be used for controller names	17
Rule: Variable path segments may be substituted with identity-based values	18
Rule: CRUD function names should not be used in URIs	18
URI Query Design	19
Rule: The query component of a URI may be used to filter collections or stores	19
Rule: The query component of a URI should be used to paginate collection or store results	20
Recap	20
<b>3. Interaction Design with HTTP .....</b>	<b>23</b>
HTTP/1.1	23
Request Methods	23
Rule: GET and POST must not be used to tunnel other request methods	24
Rule: GET must be used to retrieve a representation of a resource	24
Rule: HEAD should be used to retrieve response headers	25
Rule: PUT must be used to both insert and update a stored resource	25
Rule: PUT must be used to update mutable resources	26
Rule: POST must be used to create a new resource in a collection	26
Rule: POST must be used to execute controllers	26
Rule: DELETE must be used to remove a resource from its parent	27
Rule: OPTIONS should be used to retrieve metadata that describes a resource's available interactions	27
Response Status Codes	28
Rule: 200 ("OK") should be used to indicate nonspecific success	28
Rule: 200 ("OK") must not be used to communicate errors in the response body	28
Rule: 201 ("Created") must be used to indicate successful resource creation	28

Rule: 202 (“Accepted”) must be used to indicate successful start of an asynchronous action	29
Rule: 204 (“No Content”) should be used when the response body is intentionally empty	29
Rule: 301 (“Moved Permanently”) should be used to relocate resources	29
Rule: 302 (“Found”) should not be used	29
Rule: 303 (“See Other”) should be used to refer the client to a different URI	30
Rule: 304 (“Not Modified”) should be used to preserve bandwidth	30
Rule: 307 (“Temporary Redirect”) should be used to tell clients to re-submit the request to another URI	30
Rule: 400 (“Bad Request”) may be used to indicate nonspecific failure	30
Rule: 401 (“Unauthorized”) must be used when there is a problem with the client’s credentials	31
Rule: 403 (“Forbidden”) should be used to forbid access regardless of authorization state	31
Rule: 404 (“Not Found”) must be used when a client’s URI cannot be mapped to a resource	31
Rule: 405 (“Method Not Allowed”) must be used when the HTTP method is not supported	31
Rule: 406 (“Not Acceptable”) must be used when the requested media type cannot be served	32
Rule: 409 (“Conflict”) should be used to indicate a violation of resource state	32
Rule: 412 (“Precondition Failed”) should be used to support conditional operations	32
Rule: 415 (“Unsupported Media Type”) must be used when the media type of a request’s payload cannot be processed	32
Rule: 500 (“Internal Server Error”) should be used to indicate API malfunction	32
Recap	33

<b>4. Metadata Design</b> .....	<b>35</b>
HTTP Headers	35
Rule: Content-Type must be used	35
Rule: Content-Length should be used	35
Rule: Last-Modified should be used in responses	35
Rule: ETag should be used in responses	36
Rule: Stores must support conditional PUT requests	36
Rule: Location must be used to specify the URI of a newly created resource	37
Rule: Cache-Control, Expires, and Date response headers should be used to encourage caching	37

Rule: Cache-Control, Expires, and Pragma response headers may be used to discourage caching	38
Rule: Caching should be encouraged	38
Rule: Expiration caching headers should be used with 200 (“OK”) responses	38
Rule: Expiration caching headers may optionally be used with 3xx and 4xx responses	38
Rule: Custom HTTP headers must not be used to change the behavior of HTTP methods	38
Media Types	39
Media Type Syntax	39
Registered Media Types	39
Vendor-Specific Media Types	40
Media Type Design	41
Rule: Application-specific media types should be used	41
Rule: Media type negotiation should be supported when multiple representations are available	43
Rule: Media type selection using a query parameter may be supported	44
Recap	44

## 5. Representation Design ..... 47

Message Body Format	47
Rule: JSON should be supported for resource representation	47
Rule: JSON must be well-formed	48
Rule: XML and other formats may optionally be used for resource representation	48
Rule: Additional envelopes must not be created	49
Hypermedia Representation	49
Rule: A consistent form should be used to represent links	49
Rule: A consistent form should be used to represent link relations	52
Rule: A consistent form should be used to advertise links	53
Rule: A self link should be included in response message body representations	54
Rule: Minimize the number of advertised “entry point” API URIs	54
Rule: Links should be used to advertise a resource’s available actions in a state-sensitive manner	55
Media Type Representation	56
Rule: A consistent form should be used to represent media type formats	56
Rule: A consistent form should be used to represent media type schemas	59
Error Representation	68
Rule: A consistent form should be used to represent errors	68
Rule: A consistent form should be used to represent error responses	69

Rule: Consistent error types should be used for common error conditions	70
Recap	70
<b>6. Client Concerns</b>	<b>71</b>
Introduction	71
Versioning	71
Rule: New URIs should be used to introduce new concepts	71
Rule: Schemas should be used to manage representational form versions	72
Rule: Entity tags should be used to manage representational state versions	72
Security	72
Rule: OAuth may be used to protect resources	72
Rule: API management solutions may be used to protect resources	73
Response Representation Composition	73
Rule: The query component of a URI should be used to support partial responses	74
Rule: The query component of a URI should be used to embed linked resources	76
Processing Hypermedia	78
JavaScript Clients	79
Rule: JSONP should be supported to provide multi-origin read access from JavaScript	80
Rule: CORS should be supported to provide multi-origin read/write access from JavaScript	82
Recap	83
<b>7. Final Thoughts</b>	<b>85</b>
State of the Art	85
Uniform Implementation	86
Principle: REST API designs differ more than necessary	86
Principle: A REST API should be designed, not coded	87
Principle: Programmers and their organizations benefit from consistency	88
Principle: A REST API should be created using a GUI tool	89
Recap	91
<b>Appendix: My First REST API</b>	<b>93</b>