|  |
| --- |
| Infosys Technologies Limited |
| Best Practices on Restful API Design |
| [Type the document subtitle] |
|  |
| **Ritwik\_Chatterjee** |
| **5/31/2014** |

|  |
| --- |
| This documents describes the principles and best practices for designing Restful APIs. |

**Version**

|  |  |  |  |
| --- | --- | --- | --- |
| Version No. | Date | Created By | Comments |
| 0.10 | 31-May-2014 | Ritwik Chatterjee | Initial Draft |
|  |  |  |  |
|  |  |  |  |

**Principles**

* Don’t make your developers think. It should be intuitive
* Resource-Verbs and collections
* Use HTTP verbs to perform actions
* Collections
  + /v1/resource**s – plural to show case collection**
* Versioning
  + Version changes should only be when the interface to developers has changed. Changes to implementation/back-end should not change the API version – versioning should be on the original contract with the developer
  + Version in the URL
* Conveniences
  + Things that make it easier for the developer
  + For e.g. /facebook/me
  + **Keep it consistent**
* Payload format
  + JSON preferred format
  + Need to support multiple format (should not force developers to use to a particular format)
  + **Be consistent**
  + Ways to specify format (in request)
    - /<resource>.<format>
    - Query string /<resource>?format=<format>
    - Content-type in header or Accept header
    - **Use only 1 of the above. Be consistent**
* Date format
  + ISO Date format : YYYY-MM-DD
  + To make it easy to do math you can use YYYYMMDD
* Fields & filter criteria
  + Use query parameters to specify fields to return
    - /<resource>?fields=<field names>[,<field names>]
  + Use query parameters to specify filters/selection criteria
    - /<resource>?[<field-1>=<criteria>][&<field-2>=<criteria>]
  + In case of complex filter criteria, allow users to specify entire query (usergrid model)
    - /<resource>?ql=Select \* where created < 20140325 and salary >15000&orderby=salary&ordertype=desc
* Action APIs
  + APIs which does an action instead of just returning a resource. For e.g. calculation
  + Be very careful on http verb usage. **Be consistent**
  + /<action noun>?<Query string on what to do>
    - /search?q=aap
    - /convert?from=USD&to=INR&amount=1000
* Pagination
  + Use parameters, for e.g.
    - Offset, limit
    - Start, count
    - Start, size
* Errors
  + Be consistent
  + Create helpful error messages
  + Don’t
    - Send multiple, conflicting error codes, for e.g. HTTP200 OK with resource Not Found (an obvious HTTP404)
    - Respond back with error message/code in default format
  + Do’s
    - Respond back with error message/code in the format requested by consumer
    - Include help URL/additional help message to fix the issue
    - Send an error code that makes most sense.
  + Acceptable response code
    - Check for the link wiki article “acceptable response codes”
    - Suggested HTTP Response Codes
      * 200 OK
      * 201 Created
      * 202 Accepted – (for back-end/long running transactions)
      * 204 No Content – (for successfully deleted)
      * 400 Bad Request
      * 401 Unauthorized
      * 403 Forbidden
      * 404 Not found
      * 429 Too Many Requests (throttle/quota limits – may be at API Manager level)
      * 500 Internal Server Error
  + Allow the developer to know what has happened –
    - for e.g. send 201 created, along with id of the record created
    - Send 204 for a delete success
    - Send 404 for a second attempt to delete as the resource was already deleted
  + Good example is that of ***box***



* Authentication
  + Basic Auth
    - Sends credentials in header
    - Can be decrypted, but still in header
  + Access Token
    - Should not be sent part of the resource URL – will be logged on some server
    - Use Bearer token
      * In the header
      * Cannot extract personal information out of it
* Create resources that describes themselves at a high level