Beautiful REST + JSON APIs

Les Hazlewood @lhazlewood
CTO, Stormpath
stormpath.com

- Identity Management and Access Control API
- Security for ***your*** applications
- User security workflows
- Security best practices
- Developer tools, SDKs, libraries

Stormpath

Outline

- APIs, REST & JSON
- REST Fundamentals
- Design

Base URL
Versioning
Resource Format
Return Values
Content Negotiation
References (Linking)
Pagination
Query Parameters
Associations

Errors
IDs
Method Overloading
Resource Expansion
Partial Responses
Caching & Etags
Security
Multi Tenancy
Maintenance

Stormpath

APIs

- Applications
- Developers
- Pragmatism over Ideology
- Adoption
- Scale

Stormpath

Why REST?

- Scalability
- Generality
- Independence
- Latency (Caching)
- Security
- Encapsulation

Why JSON?

- Ubiquity
- Simplicity
- Readability
- Scalability
- Flexibility

HATEOAS

- **H**ypermedia
- **A**s
- **T**he
- **E**ngine
- **O**f
- **A**pplication
- **S**tate

Further restriction on REST architectures.

Stormpath

REST Is Easy

Stormpath

REST Is *&@#$! Hard

(for providers)

Stormpath

REST *can* be easy

(if you follow some guidelines)

Stormpath

Example Domain: Stormpath

- Applications
- Directories
- Accounts
- Groups
- Associations
- Workflows

Fundamentals

Resources

Nouns, not Verbs

Coarse Grained, not Fine Grained

Architectural style for use-case scalability

What If?

/getAccount

/createDirectory

/updateGroup

/verifyAccountEmailAddress

Stormpath

What If?

/getAccount
/getAllAccounts
/searchAccounts
/createDirectory
/createLdapDirectory
/updateGroup
/updateGroupName
/findGroupsByDirectory
/searchGroupsByName
/verifyAccountEmailAddress
/verifyAccountEmailAddressByToken
…
Smells like bad RPC. DON’T DO THIS.

Stormpath

Keep It Simple

Stormpath

The Answer

Fundamentally two types of resources:

Collection Resource

Instance Resource

Collection Resource

`/applications`

Instance Resource

```
/applications/a1b2c3
```

Stormpath

Behavior

- GET
- PUT
- POST
- DELETE
- HEAD

Stormpath

Behavior

`POST, GET, PUT, DELETE`

≠ 1:1

Create, Read, Update, Delete

Behavior

As you would expect:

`GET` = Read

`DELETE` = Delete

`HEAD` = Headers, no Body

Behavior

Not so obvious:

`PUT` and `POST` can *both* be used for Create *and* Update

PUT for Create

Identifier is known by the client:

```
PUT /applications/clientSpecifiedId

{
   …
}
```

Stormpath

PUT for Update

Full Replacement

```
PUT /applications/existingId
{
   “name”: “Best App Ever”,
   “description”: “Awesomeness”
}
```

Stormpath

PUT

Idempotent

POST as Create

On a parent resource

```
POST /applications
{
  “name”: “Best App Ever”
}

Response:

201 Created
Location: https://api.stormpath.com/applications/a1b2c3
```

Stormpath

POST as Update

On instance resource

```
POST /applications/a1b2c3

{
  “name”: “Best App Ever. Srsly.”
}

Response:

200 OK
```

Stormpath

POST

NOT Idempotent

Media Types

- Format Specification + Parsing Rules
- Request: `Accept` header
- Response: `Content-Type` header

- `application/json`
- `application/foo+json`
- `application/foo+json;application`
- …

Stormpath

Design Time!

Base URL

http(s)://api.foo.com

vs

http://www.foo.com/dev/service/api/rest

http(s)://api.foo.com

Rest Client
vs
Browser

Versioning

URL

```
https://api.stormpath.com/v1
```

vs.

Media-Type

```
application/json+foo;application&v=1
```

Stormpath

Resource Format

Media Type

```
Content-Type: application/json
```

When time allows:

```
application/foo+json
application/foo+json;bar=baz&v=1
...
```

Stormpath

camelCase

‘JS’ in ‘JSON’ = JavaScript

`myArray.forEach`
Not `myArray.for_each`

`account.givenName`
Not `account.given_name`

Underscores for property/function names are unconventional for JS. Stay consistent.

Stormpath

Date/Time/Timestamp

There's already a standard. Use it: ISO 8601

Example:

```
{
  …,
  "createdTimestamp": "2012-07-10T18:02:24.343Z"
}
```

Use UTC!

Stormpath

HREF

- Distributed Hypermedia is paramount!
- Every accessible Resource has a unique URL.
- Replaces IDs (IDs exist, but are opaque).

```
{
  “href”: https://api.stormpath.com/v1/accounts/x7y8z9”,
  …
}
```

Critical for linking, as we’ll soon see

Stormpath

Response Body

GET obvious

What about POST?

Return the representation in the response when feasible.

Add override (?_body=false) for control

Stormpath

Content Negotiation

Header

- `Accept` header

- Header values comma delimited in order of preference

```
GET /applications/a1b2c3
Accept: application/json, text/plain
```

Stormpath

Resource Extension

```
/applications/a1b2c3.json
/applications/a1b2c3.csv
```

…

Conventionally overrides `Accept` header

Resource References
aka ‘Linking’

- Hypermedia is paramount.
- Linking is fundamental to scalability.

- Tricky in JSON
- XML has it (XLink), JSON doesn’t
- How do we do it?

Instance Reference

```
GET /accounts/x7y8z9
```

```
200 OK
{
  “href”: “https://api.stormpath.com/v1/accounts/x7y8z9”,
  “givenName”: “Tony”,
  “surname”: “Stark”,
  …,
  “directory”: ????
}
```

Stormpath

Instance Reference

```
GET /accounts/x7y8z9
```

```
200 OK
{
  “href”: “https://api.stormpath.com/v1/accounts/x7y8z9”,
  “givenName”: “Tony”,
  “surname”: “Stark”,
  …,
  “directory”: {
    “href”: “https://api.stormpath.com/v1/directories/g4h5i6”
  }
}
```

Stormpath

Collection Reference

```
GET /accounts/x7y8z9
```

```
200 OK
{
  “href”: “https://api.stormpath.com/v1/accounts/x7y8z9”,
  “givenName”: “Tony”,
  “surname”: “Stark”,
  …,
  “groups”: {
    “href”: “https://api.stormpath.com/v1/accounts/x7y8z9/groups”
  }
}
```

Stormpath

Reference Expansion

(aka Entity Expansion, Link Expansion)

Stormpath

Account and its Directory?

GET /accounts/x7y8z9?expand=directory

```
200 OK
{
  “href”: “https://api.stormpath.com/v1/accounts/x7y8z9”,
  “givenName”: “Tony”,
  “surname”: “Stark”,
  …,
  “directory”: {
    “href”: “https://api.stormpath.com/v1/directories/g4h5i6”,
    “name”: “Avengers”,
    “creationDate”: “2012-07-01T14:22:18.029Z”,
    …
  }
}
```

Stormpath

Partial Representations

```
GET /accounts/x7y8z9?
fields=givenName,surname,directory(name)
```

Stormpath

Pagination

Collection Resource supports query params:

- Offset
- Limit

…/applications?offset=50&limit=25

GET /accounts/x7y8z9/groups

```
200 OK
{
  “href”: “…/accounts/x7y8z9/groups”,
  “offset”: 0,
  “limit”: 25,
  “first”: { “href”: “…/accounts/x7y8z9/groups?offset=0” },
  “previous”: null,
  “next”: { “href”: “…/accounts/x7y8z9/groups?offset=25” },
  “last”: { “href”: “…”},
  “items”: [
    {
      “href”: “…”
    },
    {
      “href”: “…”
    },
    …
  ]
}
```

Stormpath

Many To Many

Group to Account

- A group can have many accounts
- An account can be in many groups
- Each mapping is a resource:

`GroupMembership`

GET /groupMemberships/23lk3j2j3

```
200 OK
{
  “href”: “…/groupMemberships/23lk3j2j3”,
  “account”: {
    “href”: “…”
  },
  “group”: {
    “href”: “…”
  },
  …
}
```

Stormpath

GET /accounts/x7y8z9

```
200 OK
{
  “href”: “…/accounts/x7y8z9”,
  “givenName”: “Tony”,
  “surname”: “Stark”,
  …,
  “groups”: {
    “href”: “…/accounts/x7y8z9/groups”
  },
  “groupMemberships”: {
    “href”: “…/groupMemberships?accountId=x7y8z9”
  }
}
```

Stormpath

Errors

- As descriptive as possible
- As much information as possible
- Developers are your customers

POST /directories

```
409 Conflict
{
  “status”: 409,
  “code”: 40924,
  “property”: “name”,
  “message”: “A Directory named ‘Avengers’ already exists.”,
  “developerMessage”: “A directory named ‘Avengers’ already exists.  If you have a stale local cache, please expire it now.”,
  “moreInfo”: “https://www.stormpath.com/docs/api/errors/40924”
}
```

Stormpath

Security

Avoid sessions when possible

- Authenticate every request if necessary
- Stateless

Authorize based on resource content, NOT URL!

Use Existing Protocol:

- Oauth 1.0a, Oauth2, Basic over SSL only

Custom Authentication Scheme:

- Only if you provide client code / SDK
- Only if you really, *really* know what you're doing

Use API Keys instead of Username/Passwords

Stormpath

401 vs 403

- 401 “Unauthorized” *really* means Unauthenticated

  “You need valid credentials for me to respond to this request”

- 403 “Forbidden” *really* means Unauthorized

  “I understood your credentials, but so sorry, you’re not allowed!”

Stormpath

HTTP Authentication Schemes

- Server response to issue challenge:

`WWW-Authenticate`: <*scheme name*>
`realm=`"Application Name"

- Client request to submit credentials:

`Authorization`: <*scheme name*> <*data*>

Stormpath

API Keys

- Entropy
- Password Reset
- Independence
- Speed
- Limited Exposure
- Traceability

Stormpath

IDs

- IDs should be opaque
- Should be globally unique
- Avoid sequential numbers (contention, fusking)
- Good candidates: UUIDs, ‘Url64’

HTTP Method Overrides

```
POST /accounts/x7y8z9?_method=DELETE
```

Caching &
Concurrency Control

Server (initial response):

```
ETag: "686897696a7c876b7e”
```

Client (later request):

```
If-None-Match: "686897696a7c876b7e”
```

Server (later response):

```
304 Not Modified
```

Maintenance

Use HTTP Redirects

Create abstraction layer / endpoints when migrating

Use well defined custom Media Types

- **Free** for any application
- Eliminate months of development
- Automatic security best practices

Sign Up **Now**: Stormpath.com

Stormpath