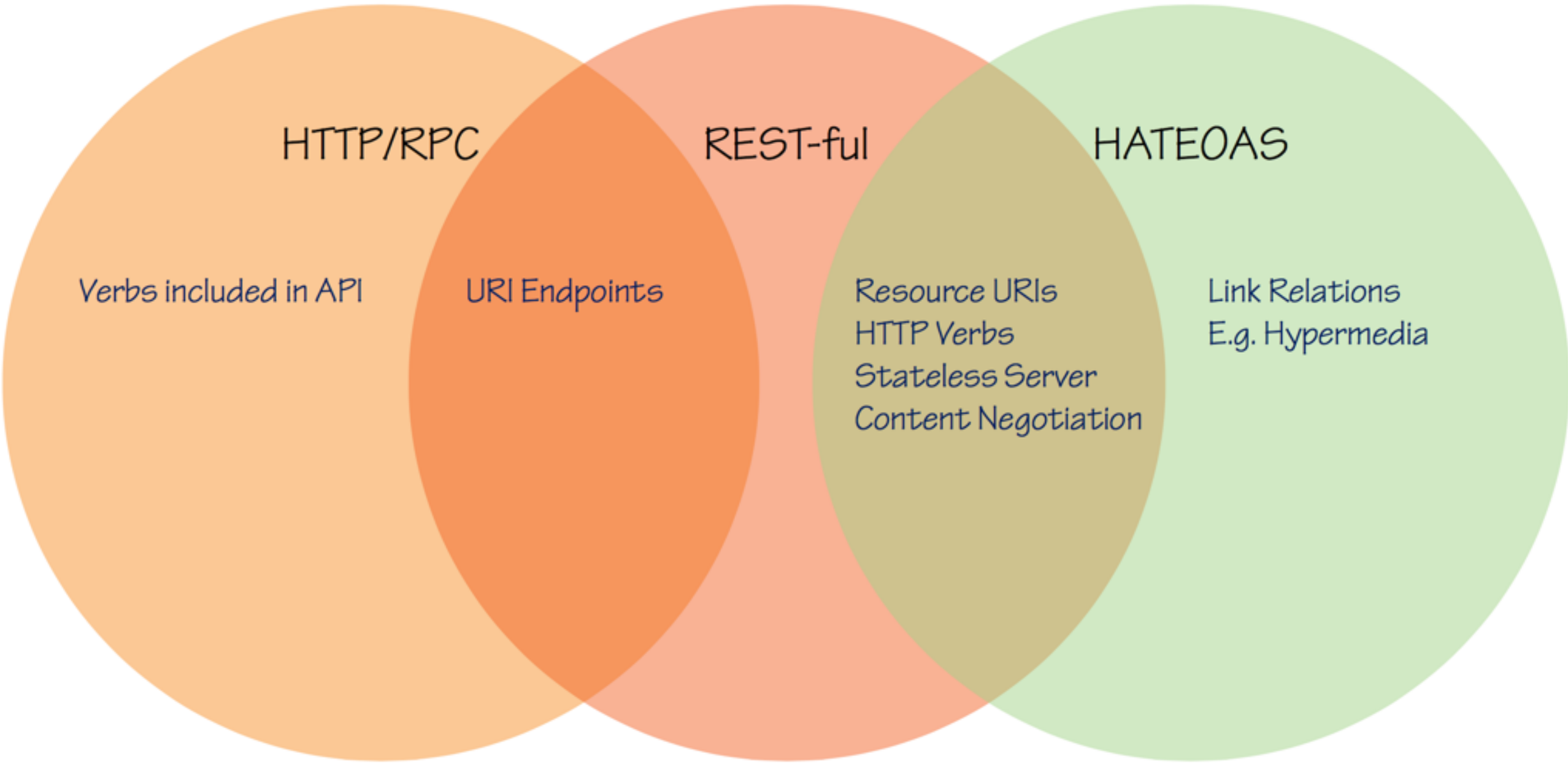
BEST PRACTICES FOR DESIGNING SUPERIOR RESTFUL APIS

Rohit Bhardwaj
Principal Cloud Engineer
rbhardwaj@kronos.com
Twitter: rbhardwaj1

DESIGNING RESTFUL APIS

- RESTful APIs
- Hypermedia
- Principles for designing APIs
- SAML and OAuth
- Customer driven contract
- API testing

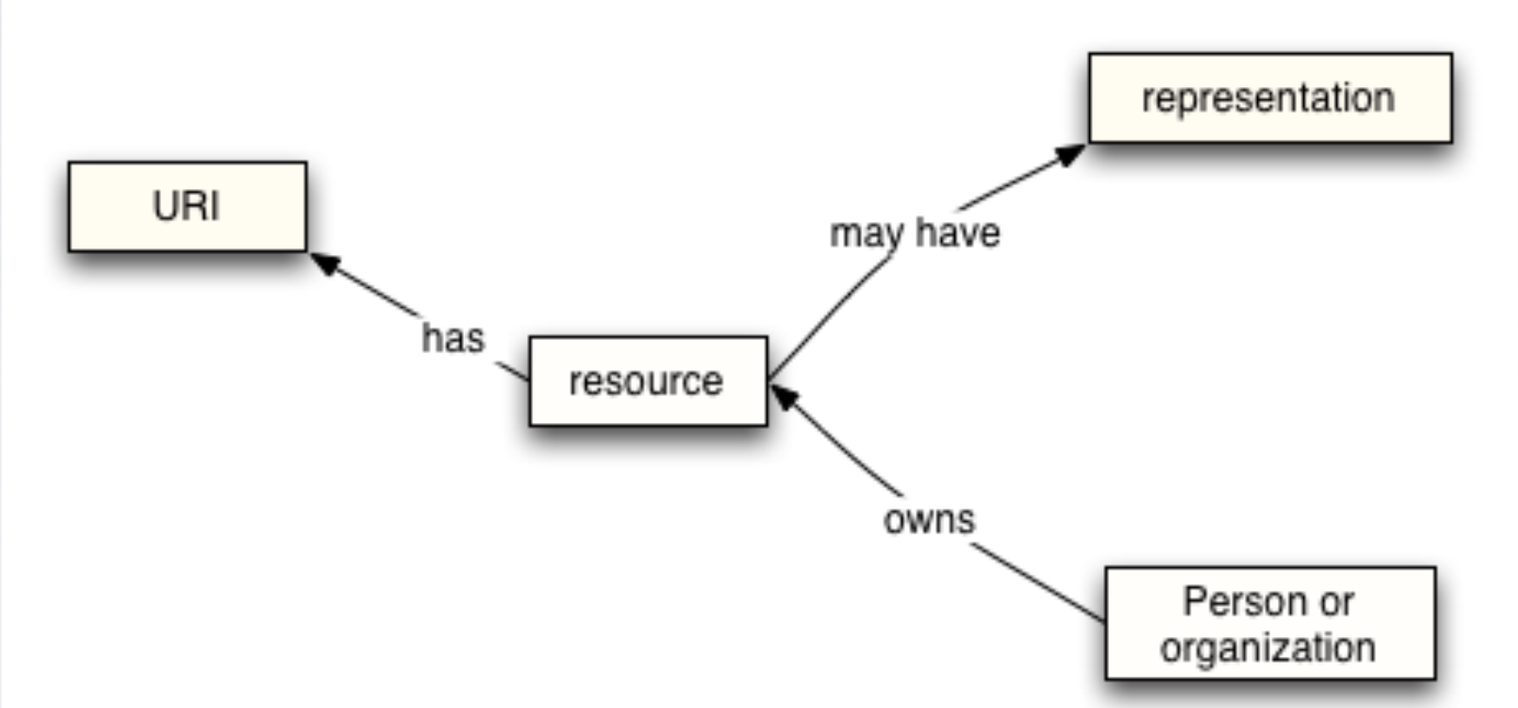
WEB API?


HTTP/RPC
Verbs included in API
REST-ful
URI Endpoints
HATEOAS
Resource URIs
HTTP Verbs
Stateless Server
Content Negotiation
Link Relations
E.g. Hypermedia

RESTFUL APIS

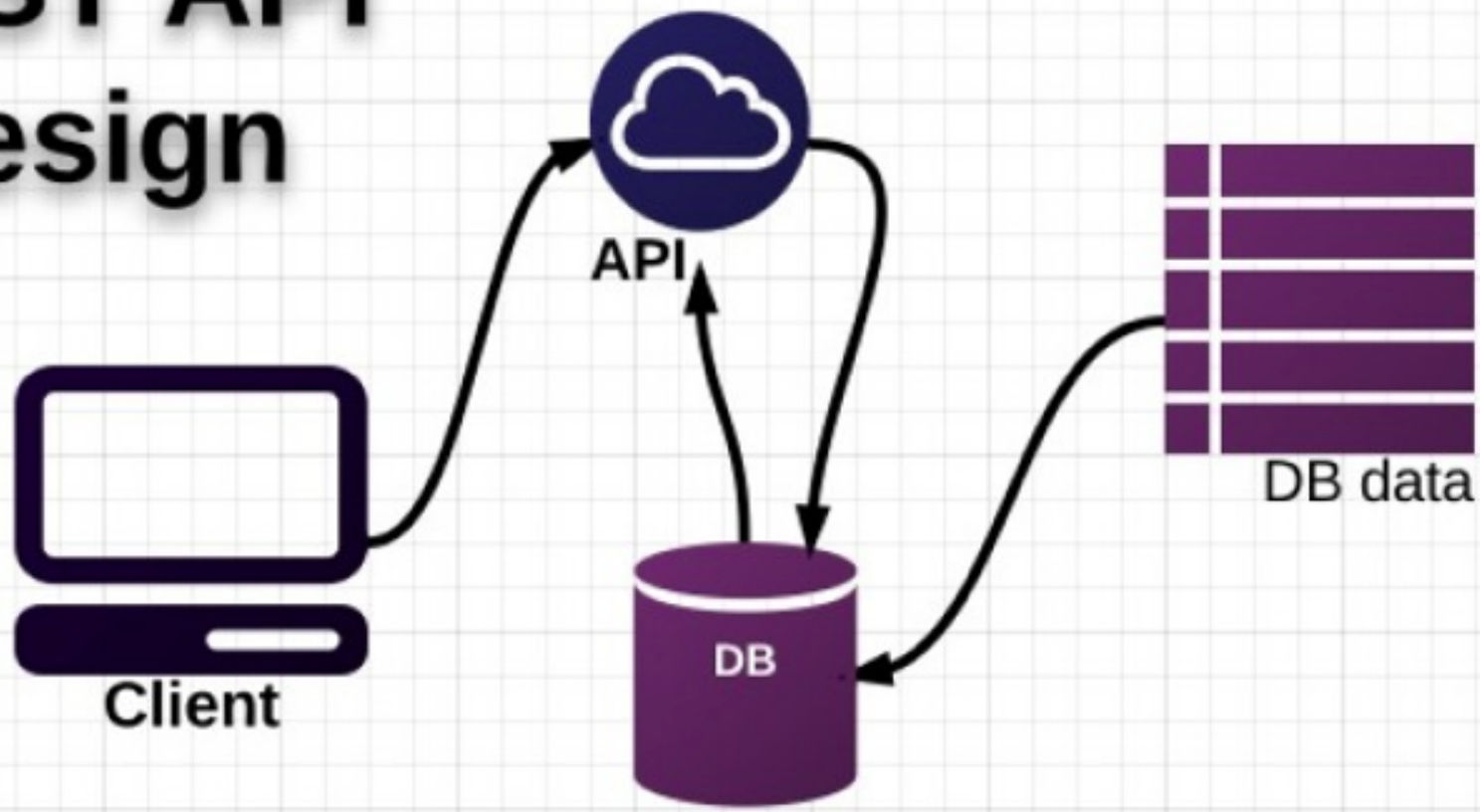
- REpresentational State Transfer
- Separation of Client and Server
- Server Requests are Stateless
- Cacheable Requests
- Uniform interface

RESOURCE BASED ARCHITECTURE



REST API Design

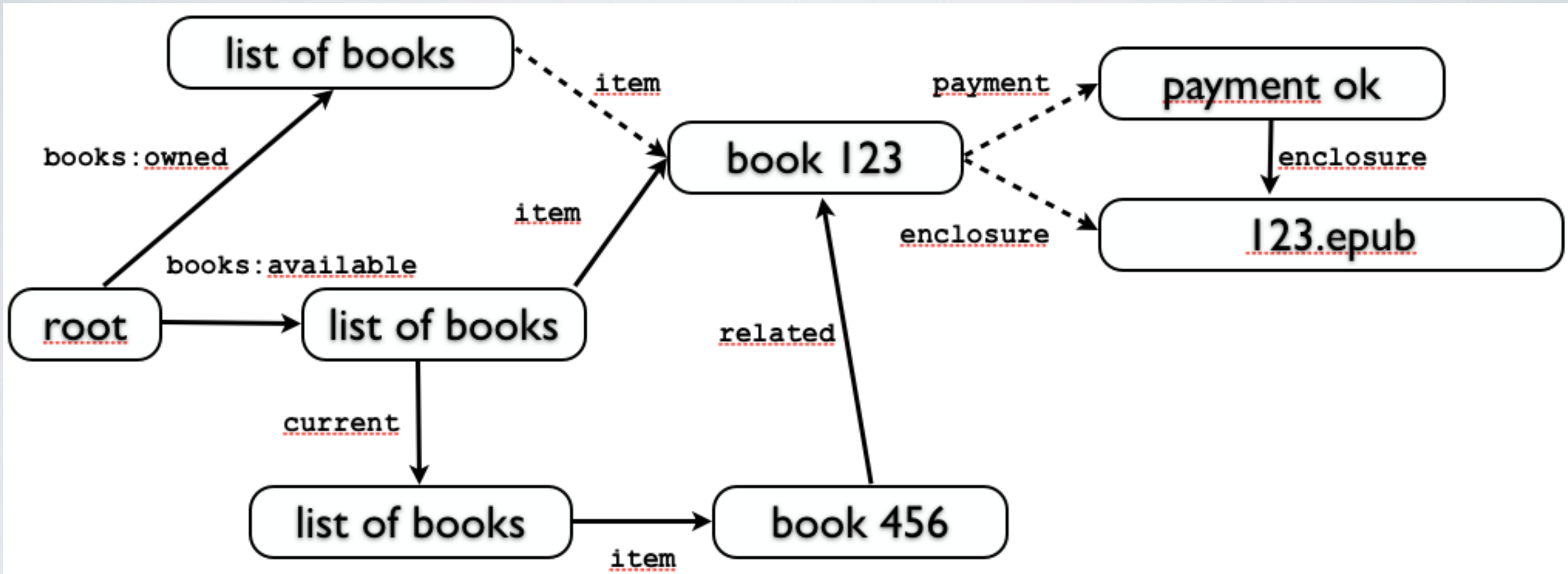
GET	/tasks - display all tasks
POST	/tasks - create a new task
GET	/tasks/{id} - display a task by ID
PUT	/tasks/{id} - update a task by ID
DELETE	/tasks/{id} - delete a task by ID



```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:body pb="http://www.acme.com/phonebook">
  <pb:GetUserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
 </soap:Body>
</soap:Envelope>
```

```
http://www.acme.com/phonebook/UserDetails/12345
```

HYPERMEDIA



http://www.jayway.com/wp-content/uploads/2012/10/hypermedia-application.png

DESIGNING APIS

REST-FUL URIS AND NON-REST-FUL URIS

- /admin/updatebook.jsp?book=5
- /bookview.jsp?book=5
- /bookandreviews.jsp?book=5

NOUNS ARE GREAT! VERBS ARE BAD

/getCustomers	/saveCustomers
/getCustomersByName	/getCustomersByPhone
/getCustomersByContact	/getCustomersUsingPaging
/getNewCustomers	/getCurrentCustomers
/createNewCustomer	/deleteCustomer
/verifyCreditCredit	/removeOldInvoice
…	

WHAT IS SOLUTION?

- Prefer Plurals
- http://../api/Customers
- http://..//api/Employees
- http://..//api/States
- http://..//api/Orders

IDENTIFIERS KEY

- http://../api/Customers/978
- http://..//api/Employees/emp123
- http://..//api/States/MA
- http://..//api/Orders/234

WHERE DOES VERBS GO?

Resource	GET (read)	POST (create)	PUT (update)	DELETE (delete)
/customers	Get List	New Customer	Update Batch	Error

WHERE DOES VERBS GO?

Resource	GET (read)	POST (create)	PUT (update)	DELETE (delete)
/customers	Get List	Create Item	Update Batch	Error
/customers/123	Get Item	Error	Update Item	Delete Item

WHAT SHOULD YOU RETURN?

Resource	GET (read)	POST (insert)	PUT (update)	DELETE (delete)
/customers	List	New Item	Status Code Only	Status Code Only*
/customers/123	Item	Status Code Only*	Updated Item	Status Code Only

RESOURCE

Verb	Action
GET	Retrieve a resource
HEAD	Get the status code as for GET, without retrieving the resource
POST	Create a new resource Catch all usage in exception cases.
PUT	Update an existing resource, through complete replacement
PATCH	Partial update of a resource
DELETE	Delete an existing resource

STATUS CODES

Code	Description	Code	Description
200	OK	400	Bad Request
201	Created	401	Not Authorized
202	Accepted	403	Forbidden
302	Found	404	Not Found
304	Not Modified	405	Method Not Allowed
307	Temp Redirect	409	Conflict
308	Perm Redirect	500	Internal Error

Code	Description	Real Story
200	OK	"It Worked"
400	Bad Request	"You did bad"
500	Internal Error	"We did bad"

Code	Description
201	Created
304	Not Modified
404	Not Found
401	Unauthorized
403	Forbidden

HOW TO SATISFY ASSOCIATIONS?

- http://../api/Customers/978/Orders
- http://..//api/Employees/emp123/Address
- http://..//api/States/MA/Cities
- http://..//api/Orders/234/Details

RETURN SAME OBJECT

- http://..//api/States/MA/Cities

- http://..//api/Cities

SAME OBJECT MULTI ASSOCIATIONS

- http://../api/Customers/978/Orders
- http://../api/Customers/978/Shipments
- http://../api/Customers/978/Invoices

WHAT IF YOU HAVE COMPLEX ASSOCIATION?

- http://../api/Customers/state=MA
- http://../api/Customers/state=MA&City=Chelmsford

HOW TO FORMAT RESULTS?

- Use content Negotiation
- Accept header

GET /api/customers/987 HTTP/1.1

Accept: application/json, text/xml

Host: localhost:9876

MIME TYPES

Type	MIME Type
JSON	application/json
XML	text/xml
JSONP*	application/javascript
RSS	application/xml+rss
ATOM	application/xml+atom

FORMATING

- http://.../api/Customers?format=json
- http://.../api/Customers.json
- http://.../api/Customers format=jsonp&callback=foo

JSON EXAMPLE

```
{
     “id” : 978
     "firstName": "John",
     "lastName": "Smith",
     "age": 45,
     "address":
     {
         "streetAddress": “21 Chelmsford Street",
         "city": "New York",
         "state": "NY",
         "postalCode": "10293"
     },
     "phoneNumber":
     [
         {
           "type": "home",
           "number": "987 555-1234"
         },
         {
           "type": "fax",
           "number": "938 555-3733"
         }
     ]
}
```

MEMBER NAMES

- Use camelCasing - Objects

HANDLING ERRORS

- Learn writing code through errors
- Test Driven Development
- Troubleshooting and resolving issues

ERROR HANDLING: APPROACH I

Facebook

```
HTTP Status Code: 200
{"type" : "OauthException", "message":"(#803) Some
of the aliases you requested do not exist: foo.bar"}
```

Twilio

```
HTTP Status Code: 401
{"status" : "401", "message":"Authenticate","code":
20003, “more info": "http://www.twilio.com/docs/
errors/20003"}
```

SimpleGeo

```
HTTP Status Code: 401
{"code" : 401, "message": "Authentication Required"}
```

BE VERBOSE AND USE PLAIN LANGUAGE DESCRIPTIONS

- {"developerMessage" : "Verbose, plain language description of the problem for the app developer with hints about how to fix it.", "userMessage":"Pass this message on to the app user if needed.", "errorCode" : 12345, "more info”: "http://dev.teachdogrest.com/errors/12345"}

DESIGNING COLLECTIONS


```
{
   “numberOfResults” : 6,
   “retults”  : [
    {
     “id” : 978
     "firstName": "John",
     "lastName": "Smith",
     "age": 45,
     "address":
   {
      "streetAddress": “21 Chelmsford Street",
      "city": "New York",
      "state": "NY",
      "postalCode": "10293"
   },
   "phoneNumber":
   [
      {
        "type": "home",
        "number": "987 555-1234"
      },
      {
        "type": "fax",
        "number": "938 555-3733"
      },
   {…….. }]
}
```

ENTITY TAGS (ETAGS)

- Header to support smart server caching

- Strong and Weak Caching

```
HTTP/1.1 200 OK
Last-Modified: Tue, 12 Dec 2015 03:03:59 GMT
ETag: "10c24bc-4ab-457wer1f"
Content-Length: 12195
```

ENTITY TAGS (ETAGS)

- Header to support smart server caching
- Strong and Weak Caching

```
HTTP/1.1 200 OK
Last-Modified: Tue, 12 Dec 2015 03:03:59 GMT
ETag: "10c24bc-4ab-457wer1f"
Content-Length: 12195



GET /i/yahoo.gif HTTP/1.1
Host: us.yimg.com
If-Modified-Since: Tue, 12 Dec 2015 03:03:59 GMT
If-None-Match: "10c24bc-4ab-457e1c1f"
HTTP/1.1 304 Not Modified

Use 304 to indicate object not modified
```

ENTITY TAGS (ETAGS)

- Header to support smart server caching

- Weak Caching: Objects are semantically same

```
HTTP/1.1 200 OK
Last-Modified: Tue, 12 Dec 2015 03:03:59 GMT
ETag: W/“10c24bc-4ab-457wer1f"
Content-Length: 12195
```

PAGING TO RETURN COLLECTIONS

- Prevent to return huge amount of data
- Large result sets performance
- Query String parameter to narrow down
- GET /customers?offset=10&limit=5

FILTERING

Query Parameter	Function	Example
where	Filtering criteria	where="from=03/30/2016"
index	Start index	index=100
count	Number of items returned	count=50
sort	Sorting direction. It can be ascending or descending	sort=ascending
by	Sorting element name	by=creation_time
{attr}	Short form. Shorthand for where="{attr}=..."	from=01/20/2013

FILTERING

- GET /customer/v1/user/orders?from=01/12/2016&to=01/12/2016&sort=ascending&by=…&index=0&count=20

- The equivalent long form is:

- GET /customer/v1/user/orders?where=”from=01/12/2016 and to=01/12/2016”&sort=ascending&by=…&index=0&count=20

BATCH CRUD OPERATIONS

- multi_create
- multi_read
- multi_update
- multi_delete

I18NL10N

- Language
  - Support ISO 639 Language codes
- Currency
  - ISO 4217: Currency codes

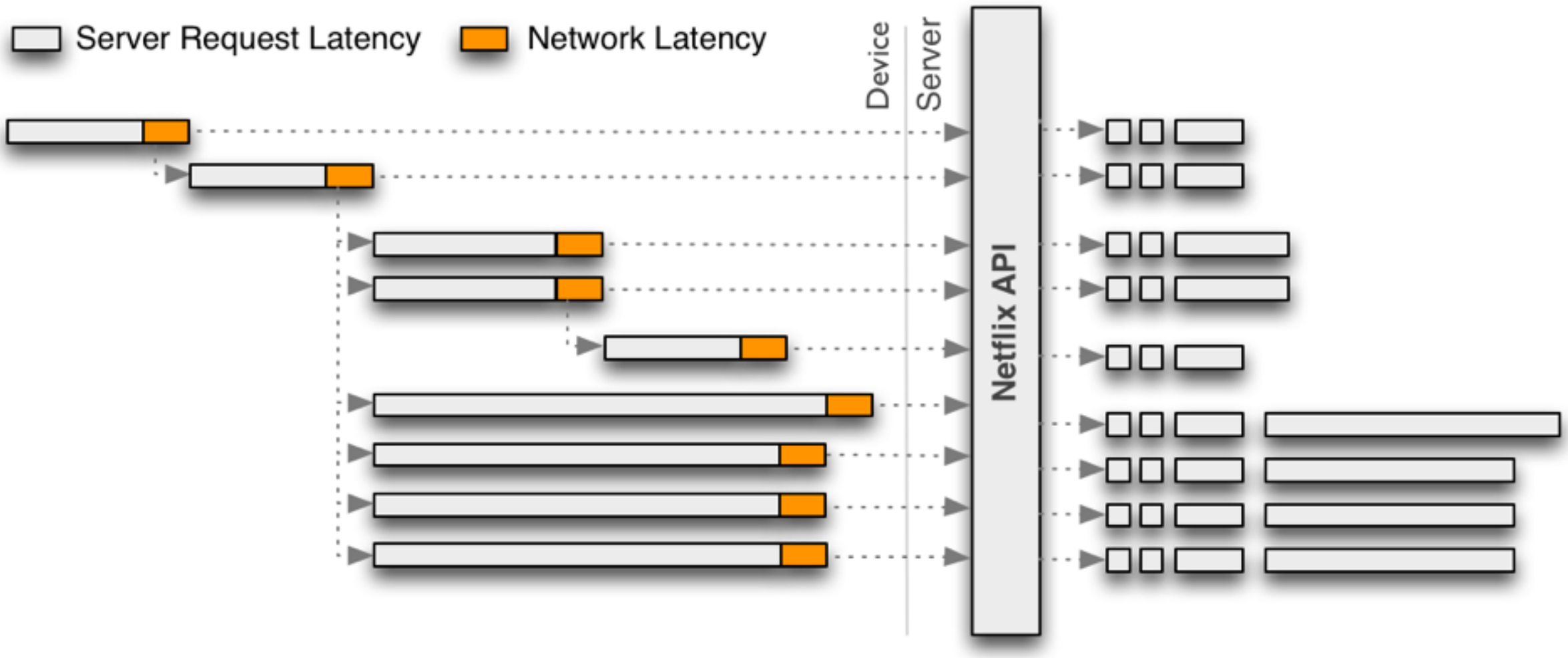
I18NL10N

- Numbers: Support Decimal XML Schema
  - +2, -123.45, 0.34, 0
- Floats and doubles : IEEE 754-1985
  - -1E4, 1234.43233E10, 12.78e-3, 13 , -0, 0

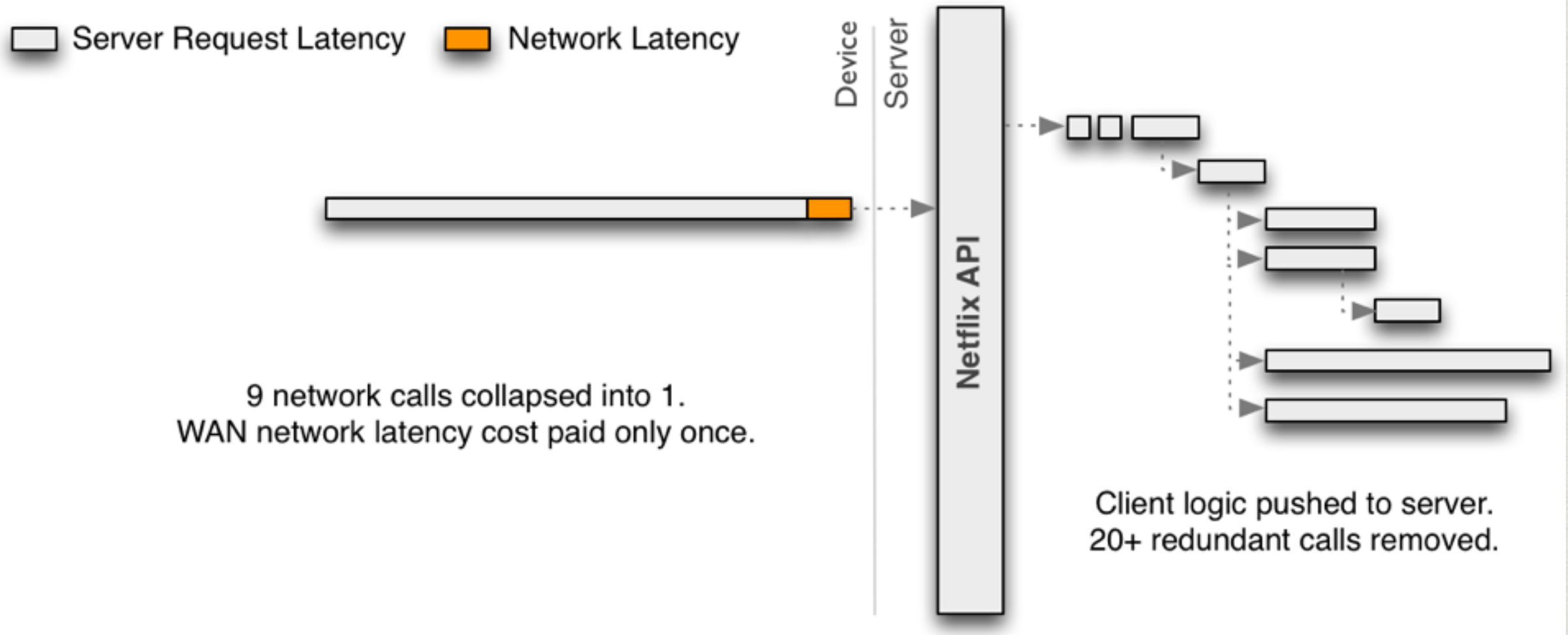
I18NL10N

- Date/time: ISO-8601 standard
  - Date: YYYY-MM-DD -> 2011-12-03
    - Date and time: YYYY-MM-DD'T'HH:MM:SS -> 2011-12-03T10:15:30
    - Time: HH:MM:SS -> 10:15:30
-

Chatty APIs: Bandwidth Aware



Reduce chattiness



http://techblog.netflix.com/2013/01/optimizing-netflix-api.html

PAGING

- {
  "totalNumResults": 9999,
  "nextPage": "http://.../api/customers/?page=5",
  "prevPage": "http://.../api/customers/?page=1",
  "results": [...]

- }

PARTIAL ITEMS

- Query string to return less
- https://www.googleapis.com/demo/v1?key=YOUR-API-KEY
- https://developers.google.com/drive/v3/web/performance#partial-response

API

- https://www.googleapis.com/demo/v1?key=YOUR-API-KEY


```
{
  "kind": "demo",
  ...
  "items": [
  {
    "title": "First title",
    "comment": "First comment.",
    "characteristics": {
      "length": "short",
      "accuracy": "high",
      "followers": ["Jo", "Will"],
    },
    "status": "active",
    ...
  },
  {
    "title": "Second title",
    "comment": "Second comment.",
    "characteristics": {
      "length": "long",
      "accuracy": "medium"
      "followers": [ ],
    },
    "status": "pending",
    ...
  },
  ...
  ]
}
```

PARTIAL ITEMS

- https://www.googleapis.com/demo/v1?key=YOUR-API-KEY&fields=kind,items(title,characteristics/length)

200 OK


```
{
  "kind": "demo",
 "items": [
 {
   "title": "First title",
   "characteristics": {
     "length": "short"
   }
 },
 {
   "title": "Second title",
   "characteristics": {
     "length": "long"
   }
 },
 ...
 ]
```

NON RESOURCE APIS

- get support only
- Should be functional API
- http://.../api/calculatePrice?state=MA&total=62.95
- http://.../api/pingServer?isNotify=true
- http://.../api/weatherForecast?isSummary=true&state=“MA”

SUMMARY

- Do not surprise users
- Follow patterns available
- Caching and use Etags
- Use of Partial items
- Non Resource APIs

VERSIONING REST APIS

WHY VERSIONING?

- Publishing an API is not a trivial move
- Users/Customers rely on the API not changing
- But requirements will change
- Need a way to evolve the API without breaking existing clients
- API Versioning isn't Product Versioning Don’t tie them together

- Thy Shall not break existing Customers

http://zentofitness.com/wp-content/uploads/2011/07/breaking-chainsS.jpg

EXAMPLES

- **Uri Path**
    - **Tumbler**
        - **http://api.tumblr.com/v2/user/**
- **Uri Parameter**
    - **Netflix**
        - **http://api.netflix.com/catalog/titles/series/70023522?v=1.5**
- **Content Negotiation**
    - **GitHub API**
        - **ContentType:application/vnd.github.1.param+json**
- **Request Header**
    - **Azure**
        - **x-ms-version:2011-08-18**

VERSIONING IN THE URI PATH

- Using Part of Your Path to Version
- Allows you to drastically change the API
- Everything below the version is open to change
- Twitter
  - http://api.twitter.com/1/users/show/noradio.xml
- Atlassian
  - http://host:port/context/rest/upm/1/plugin/a-plugin-key

URI PATH

Digg	URI	http://services.digg.com/2.0/comment.bury
Delicious	URI	https://api.del.icio.us/v1/posts/update
Last FM	URI	http://ws.audioscrobbler.com/2.0/
LinkedIn	URI	http://api.linkedin.com/v1/people/~/connections
Foursquare	URI	https://api.foursquare.com/v2/venues/40a55d80f964a52020f31ee3?oauth_token=XXX&v=YYYYMMDD
Freebase	URI	https://www.googleapis.com/freebase/v1/search?query=nirvana&indent=true

VERSIONING IN THE URI PATH

- Pro(s):
  - Simple to segregate old APIs for backwards compatibility
- Con(s):
  - Requires lots of client changes as you version
    E.g. version # has to change in every client
  - Increases the size of the URI surface area you have to support

VERSIONING IN THE URI PARAMETERS

- http://api.netflix.com/catalog/titles/series/70023522?v=1.5
- http://..//api/Employees
- http://..//api/Employees?v=3.1

VERSIONING IN THE URI PARAMETERS

- Pro(s):
  - Without version, users always get latest version of API
    - Little client change as versions mature
- Con(s):
  - Can surprise developers with unintended changes

HYPERMEDIA CONTROL

```
{
    "users": [
        {
            "displayName": "Jane Doe",
            "links": [
                { "rel": "user", "href": "/users/1234" },
                { "rel": "friends", "href": "/users/1234/friends" }
            ]
        },
        {
            "displayName": "John Doe",
            "links": [
                { "rel": "user", "href": "/users/5678" },
                { "rel": "friends", "href": "/users/5678/friends" }
            ]
        }
    ]
}
```

```
{
    "users": [
        {
            "displayName": "Jane Doe",
            "links": [
                { "rel": "user", "href": "/users/1234" },
                { "rel": "userV2", "href": "/users/v2/1234" },
                { "rel": "friends", "href": "/users/1234/friends" }
            ]
        },
        {
            "displayName": "John Doe",
            "links": [
                { "rel": "user", "href": "/users/5678" },
                { "rel": "userV2", "href": "/users/v2/5678" },
                { "rel": "friends", "href": "/users/5678/friends" }
            ]
        }
    ]
}
```

```
{
    "users": [
        {
            "displayName": "Jane Doe",
            "links": [
                { "rel": "user", "href": "/users/1234" },
                { "rel": "userV2", "href": "/users/v2/1234" },
                { "rel": "userV3", "href": "/users/v3/1234" },
                { "rel": "userV4", "href": "/users/v4/1234" },
                { "rel": "userV5", "href": "/users/v5/1234" },
                { "rel": "friends", "href": "/users/1234/friends" },
                { "rel": "friendsV2", "href": "/users/1234/V2/friends" }
            ]
        },
        {
            "displayName": "John Doe",
            "links": [
                { "rel": "user", "href": "/users/5678" },
                { "rel": "userV2", "href": "/users/v2/5678" },
                { "rel": "userV3", "href": "/users/v3/5678" },
                { "rel": "userV4", "href": "/users/v4/5678" },
                { "rel": "userV5", "href": "/users/v5/5678" },
                { "rel": "friends", "href": "/users/5678/friends" },
                { "rel": "friendsV2", "href": "/users/5678/V2/friends" }
            ]
        }
    ]
}
```

VERSIONING WITH CONTENT NEGOTIATION

- GET /api/employee/978
  HOST: http://.../
  Accept: application/myapp.**v3**.employee

- GET /api/employee/123
  HOST: http://.../
  Accept: application/myapp.**v3**.employee.json

VENDOR TYPE VERSION THE REQUEST TYPE

===>

GET /customer/123 HTTP/1.1

Accept: application/**vnd.company.myapp.customer**-v1+xml

<===

HTTP/1.1 200 OK

Content-Type: application/**vnd.company.myapp.customer**-v1+xml

<customer>

  <name>Neil Armstrong</name>

</customer>

http://thereisnorightway.blogspot.com/2011/02/versioning-and-types-in-resthttp-api.html

NEWER CLIENTS MAKE A DIFFERENT CALL

===>

GET /customer/123 HTTP/1.1

Accept: application/**vnd.company.myapp.customer**-v2+xml

<===

HTTP/1.1 200 OK

Content-Type: application/**vnd.company.myapp.customer**-v2+xml

<customer>

  <firstName<Neil>/firstName>

  <lastName<Armstrong>/lastName>

  <salutation>Mr.</salutation>

</customer>

VERSIONING WITH CONTENT NEGOTIATION

- Pro(s):
    - Packages API and Resource Versioning in one
    - Removes versioning from API so clients don't have to change
- Con(s):
    - Adds complexity - adding headers isn't easy on all platforms
    - Can encourage increased versioning which causes more code churning

VERSIONING WITH CUSTOM HEADER

GET /api/customer/123

HOST: http://.../

x-MyApp-Version: 2.1

ADDING DATE

GET /api/customer/123

HOST: http://.../

x-MyApp-Version: 2013-08-13

VERSIONING WITH CUSTOM HEADER

- Pro(s):
  - Separates Versioning from API call signatures
  - Not tied to resource versioning (e.g. Content Type) 

    Con(s):
- Adds complexity - adding headers isn't easy on all platforms

WHICH ONE TO CHOOSE?

- There is no easy answer
    - Versioning with Content Negotiation and Custom Headers are popular now
    - Versioning with URI Components are more common
        - These are easier to implement
- But you should version from the first release of your API

- YOU MUST VERSION API !

API SECURITY

SECURITY APIS

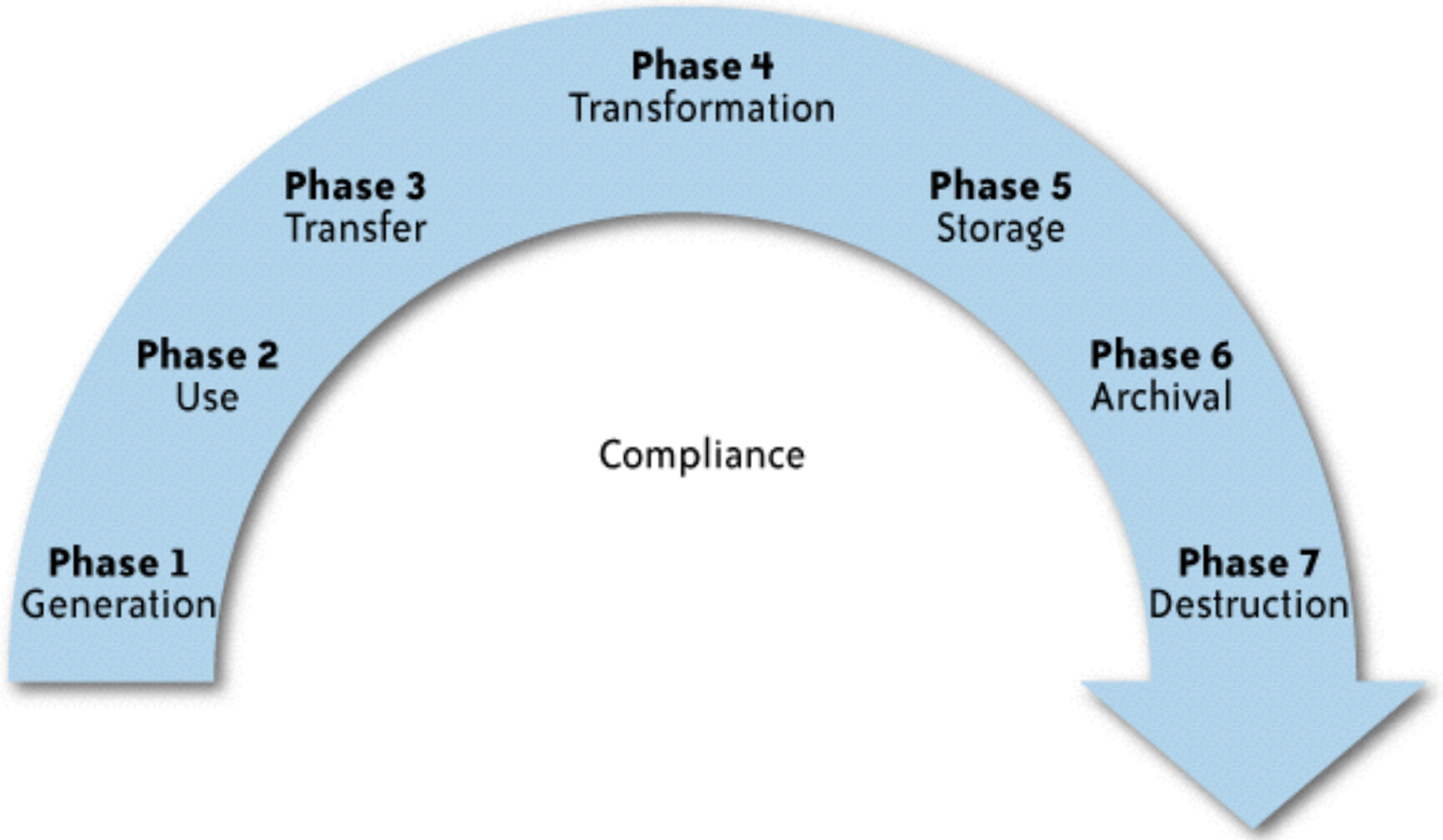
- Threats
- Protecting APIs
- Authentication
- API Keys: How they work?
- User Authentication and Authorization
- SAML
- Understanding OAuth

WHY NEED TO SECURE?

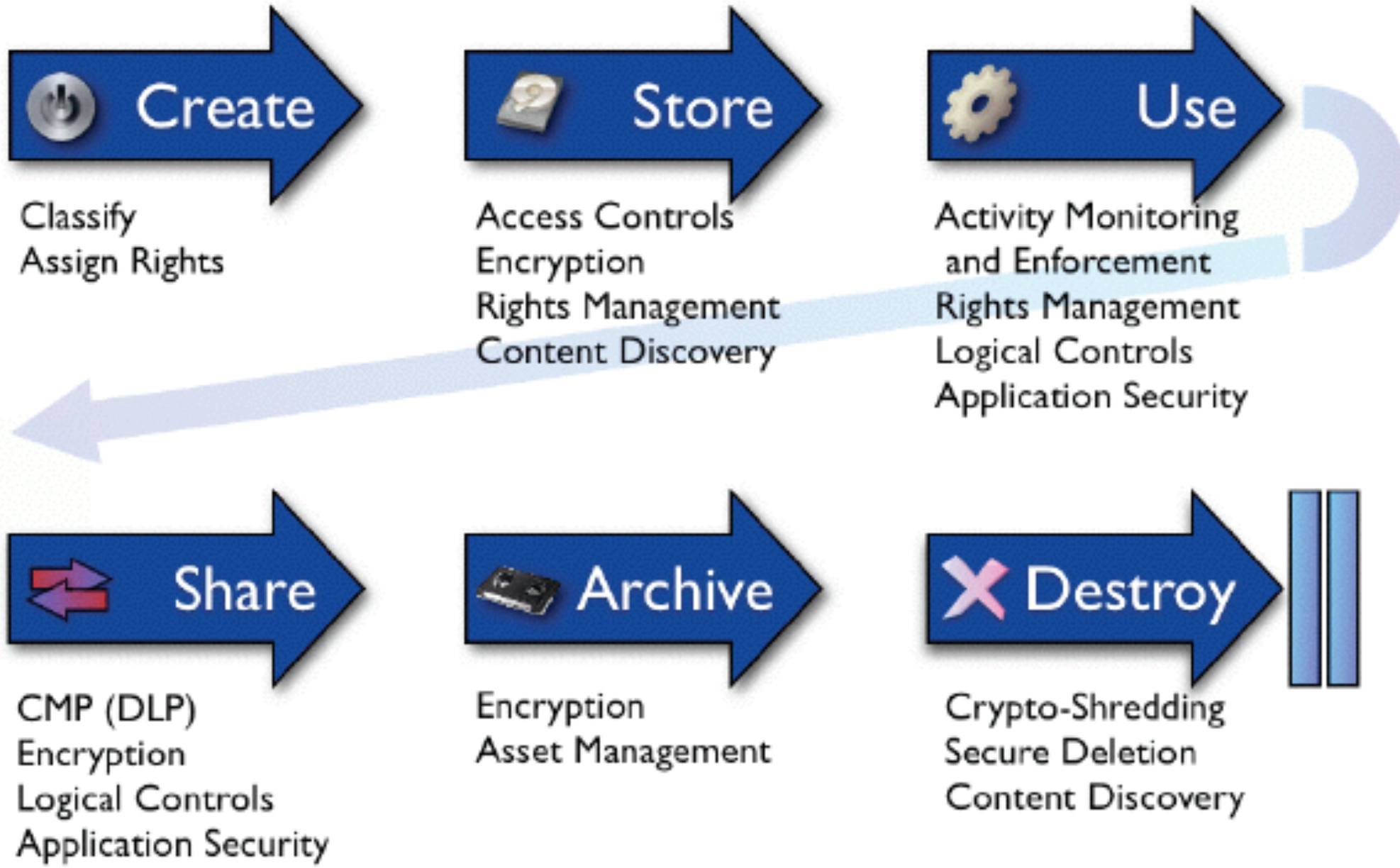
- US laws and Regulations

- International laws and regulation

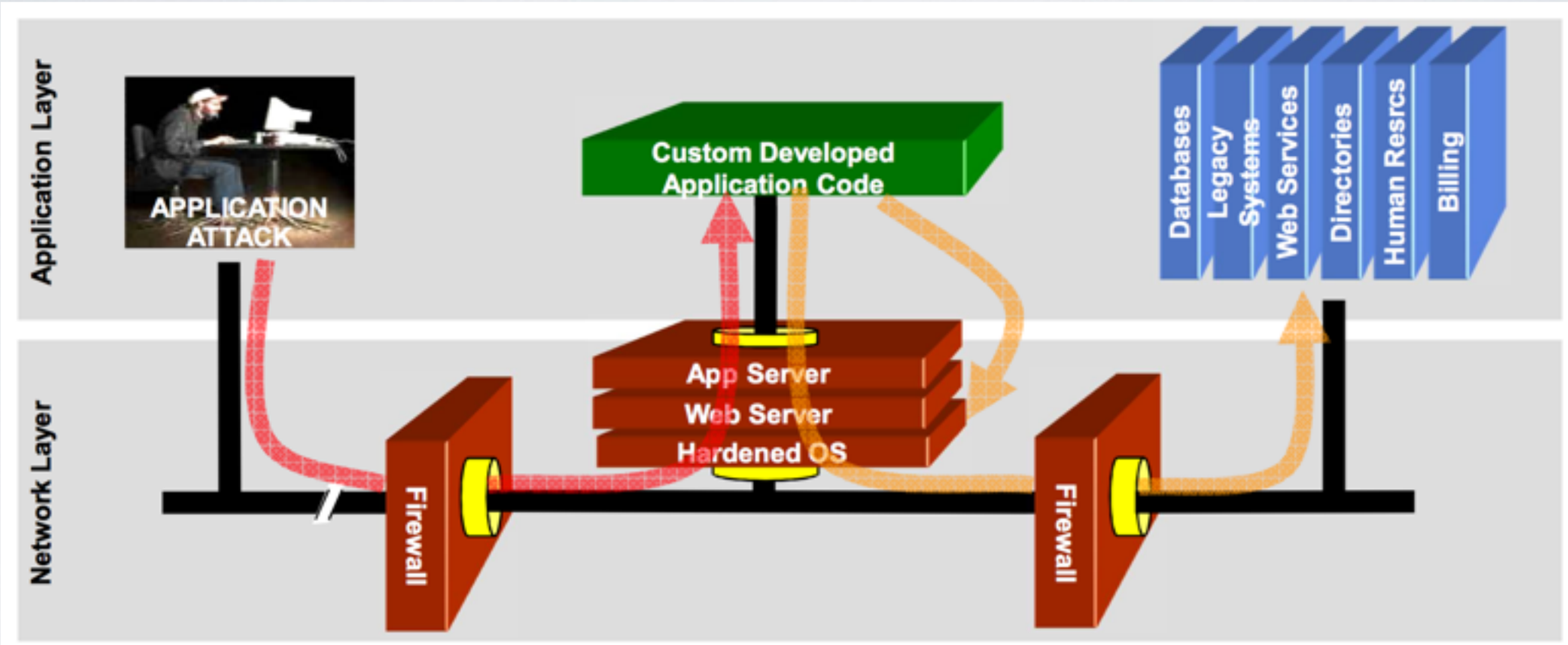
Privacy: KPMG Data life cycle



Data Security lifecycle



WHY PROTECT APIS?

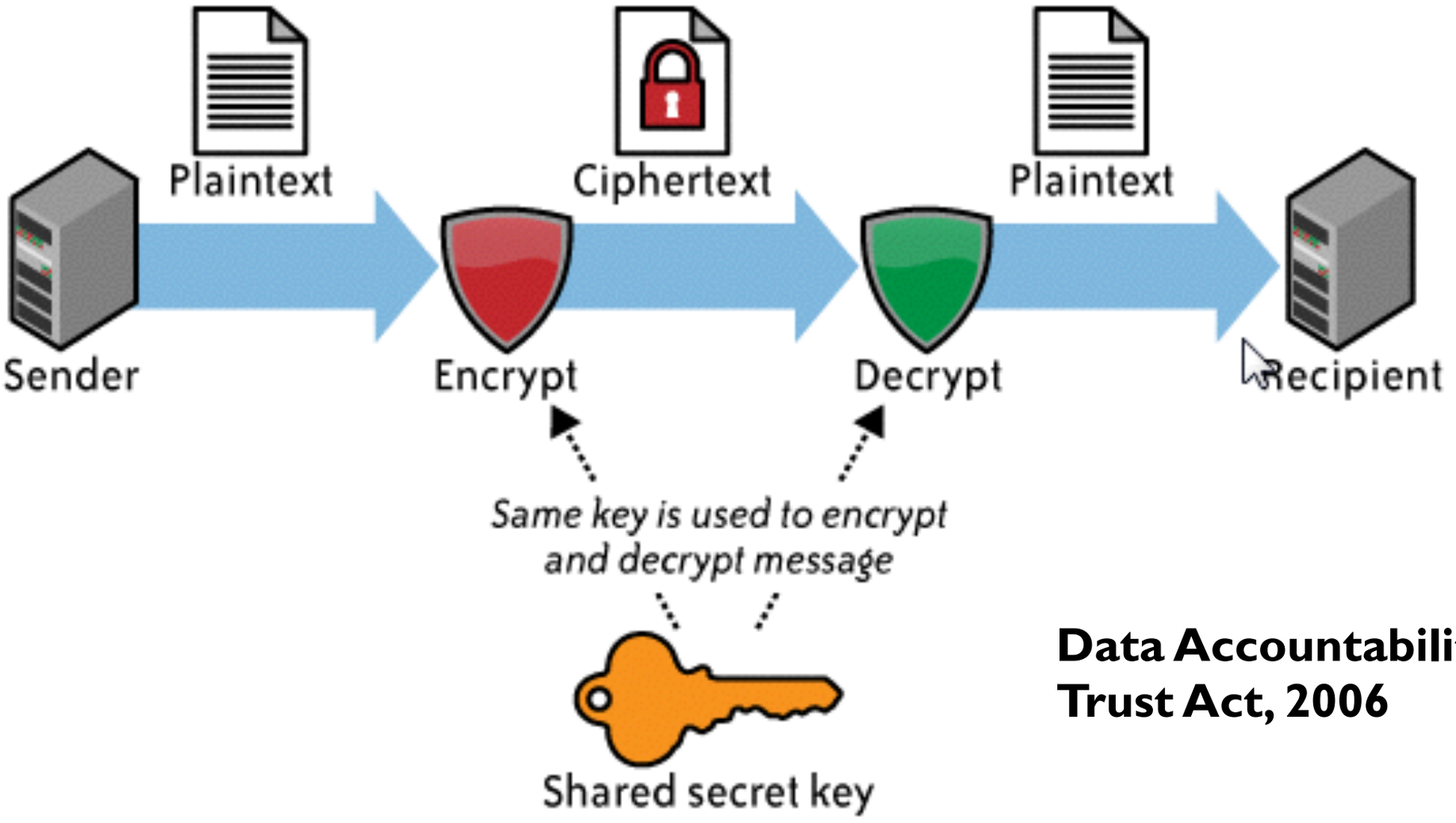


You can't use network layer protection (firewall, SSL, IDS, hardening) to stop or detect application layer attacks

- Server-to-Server Authentication
  - API Keys and Shared Secrets
- User Proxy Authentication
  - OAuth
- Direct User Authentication
  - Cookies or token

- Credential:
  - A fact that describe an identity
- Authentication:
  - Validate a set of credentials to identify an entity (whether virtual or actual)
- Authorization:
  - Verification that an entity has rights to access a resource or action

Confidentiality: Symmetric Encryption : Efficient

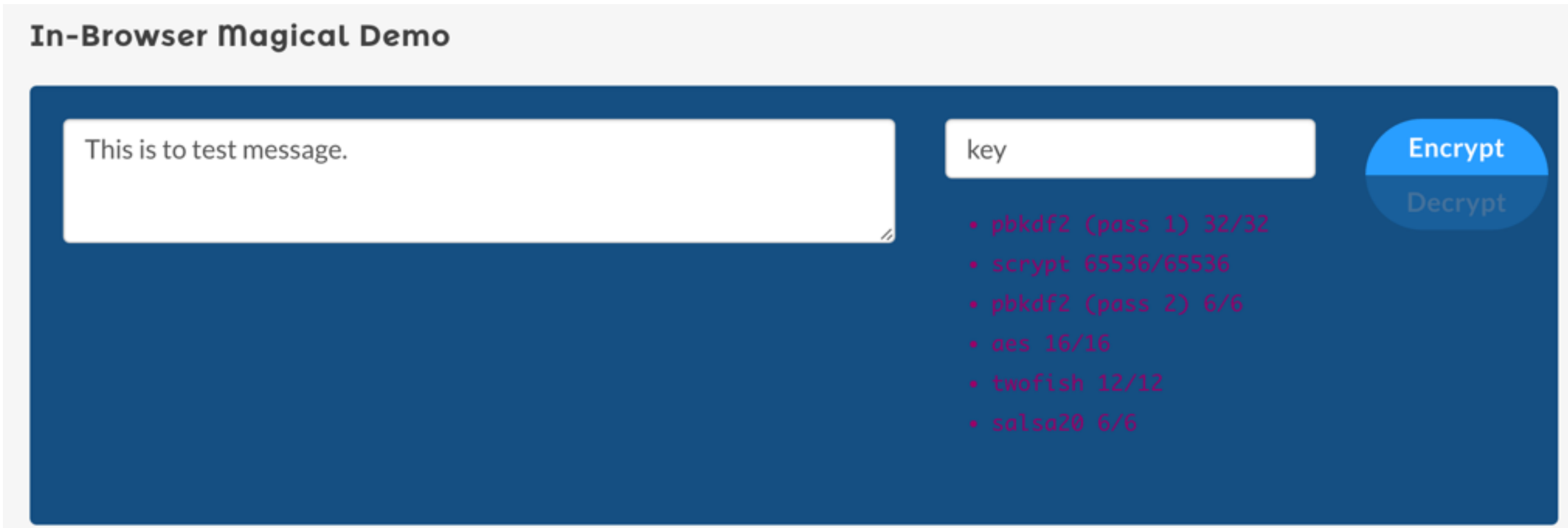


Data Accountability and Trust Act, 2006

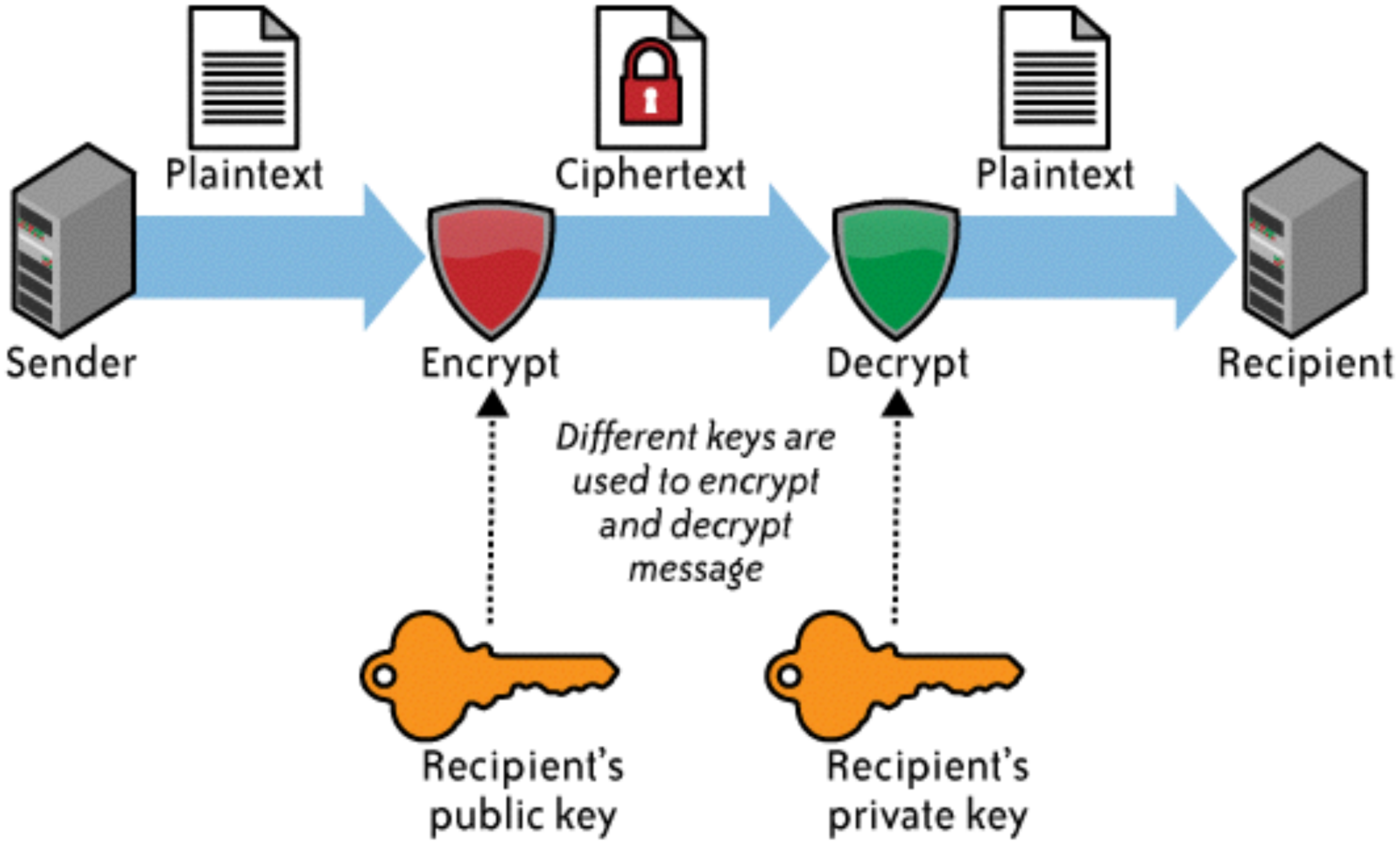
https://keybase.io/triplesec/

http://code.google.com/p/jscryptolib/

https://keybase.io/triplesec/



Confidentiality: Asymmetric Encryption



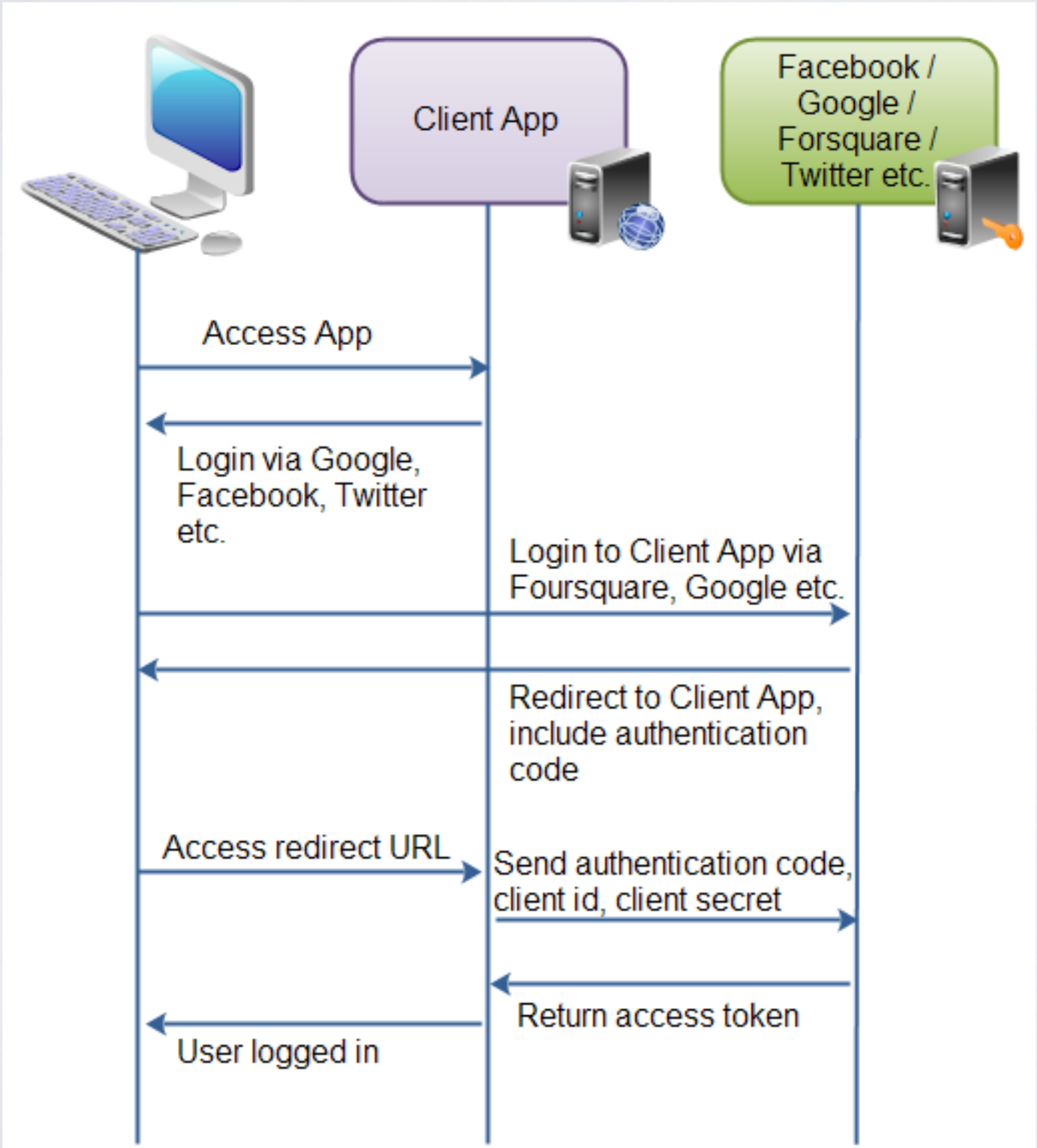
http://www.securecottage.com/demo/rsa2.html
http://support.persits.com/encrypt/demo_text.asp
http://support.persits.com/pdf/demo_encrypt.aspx

AUTHENTICATING USERS

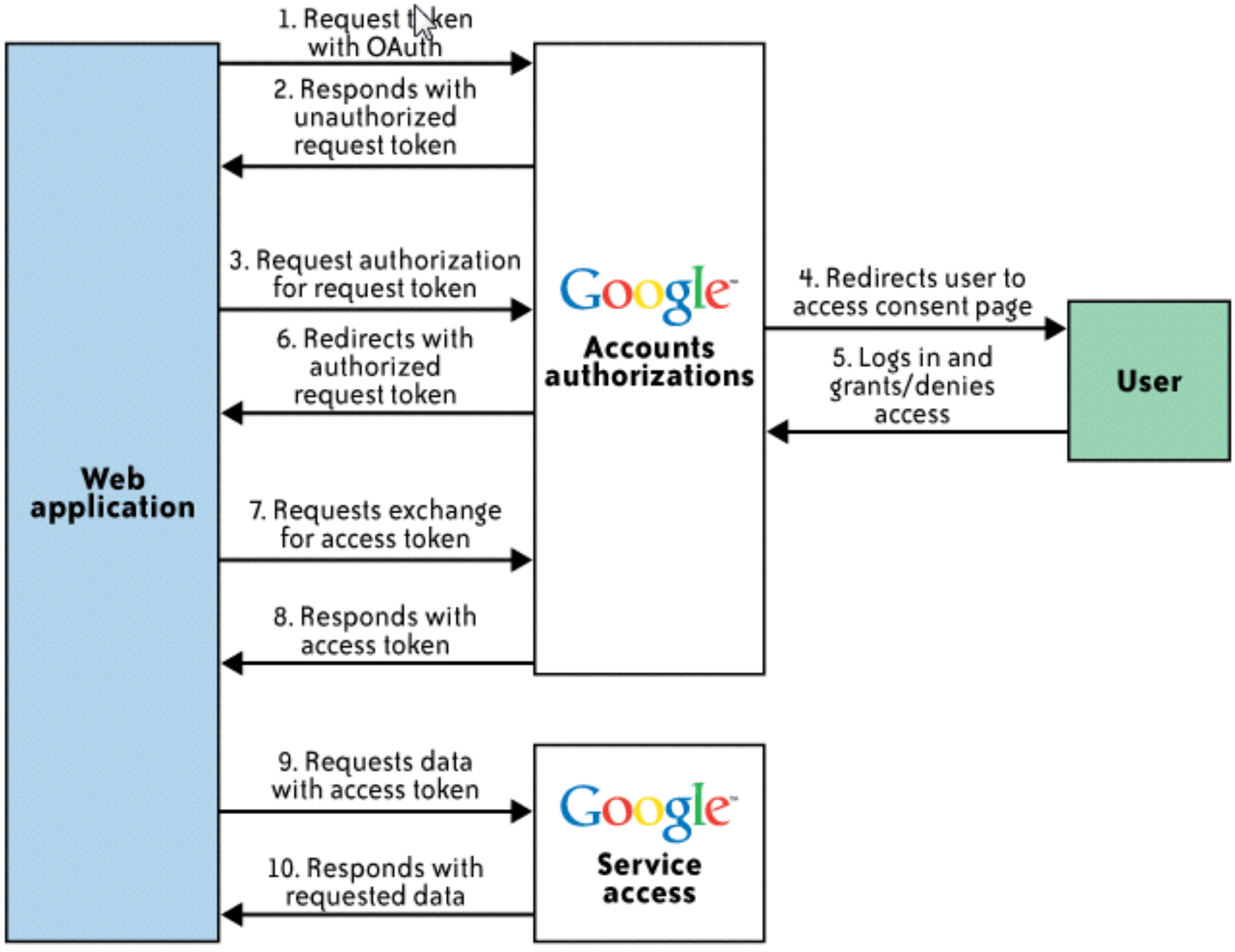
- Within system

- OAuth

OAUTH

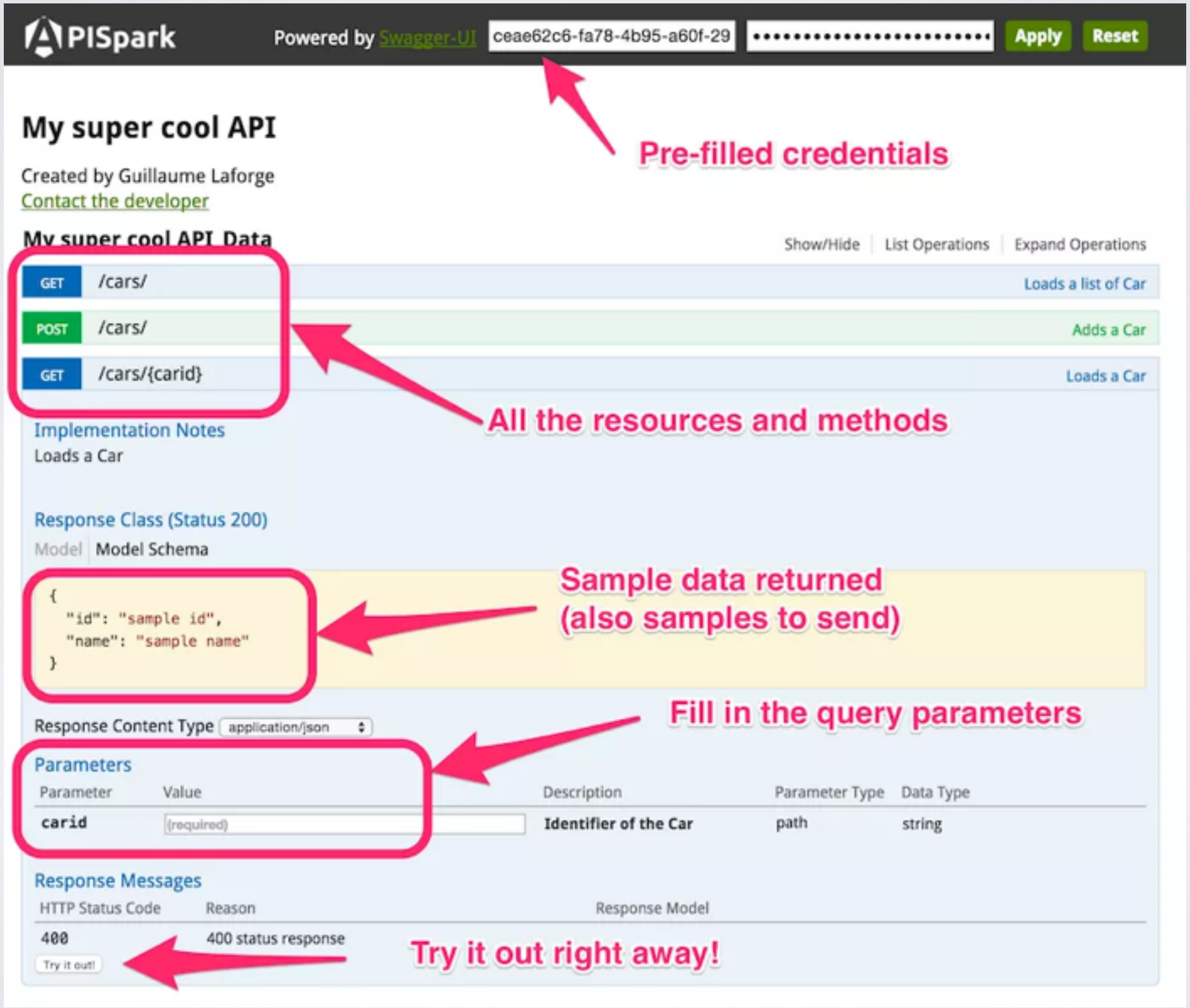

Client App
Facebook / Google / Forsquare / Twitter etc.
Access App
Login via Google, Facebook, Twitter etc.
Login to Client App via Foursquare, Google etc.
Redirect to Client App, include authentication code
Access redirect URL
Send authentication code, client id, client secret
Return access token
User logged in

Open Authentication (OAuth)

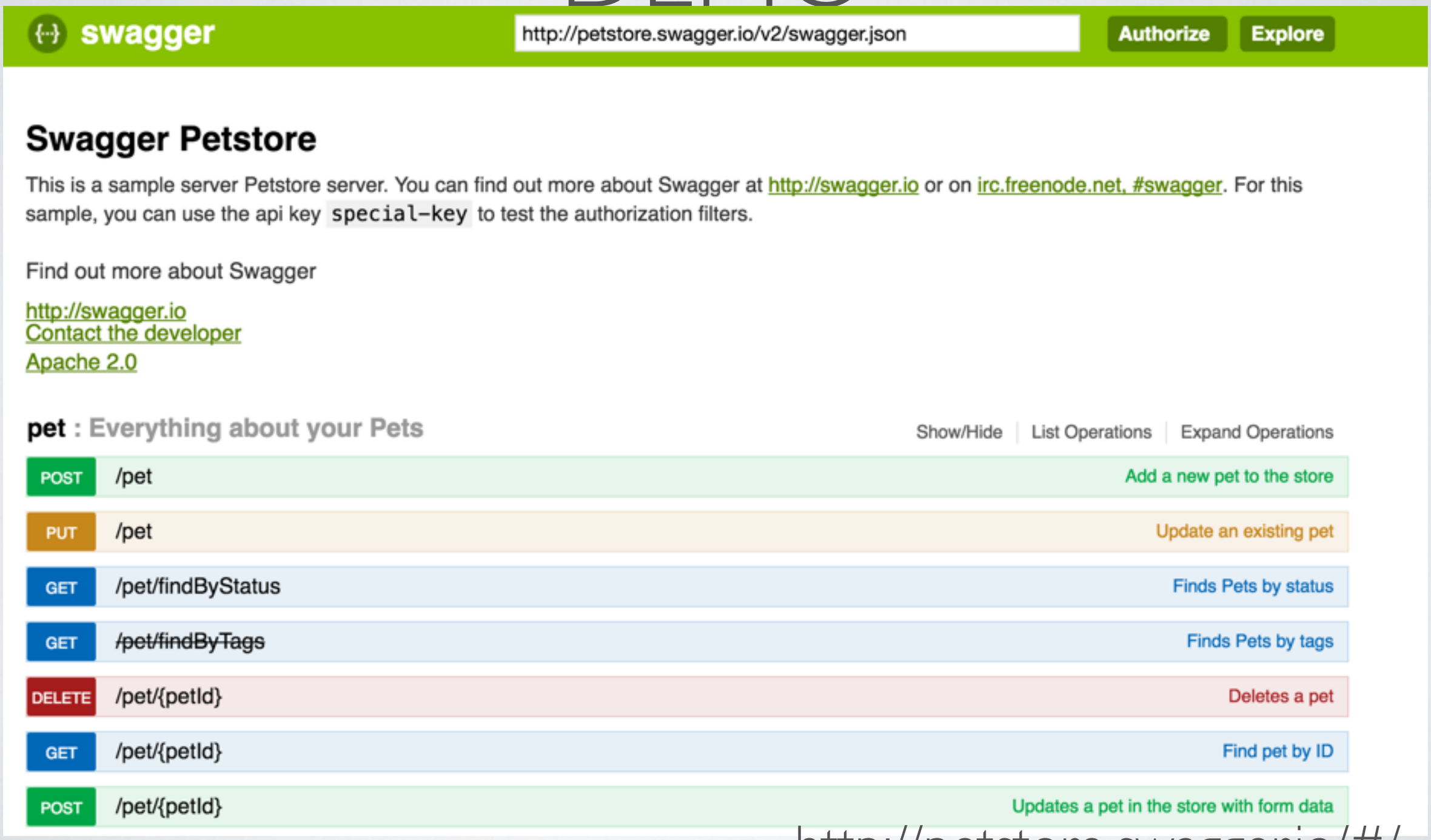


http://brentertainment.com/oauth2/

DOCUMENTATION

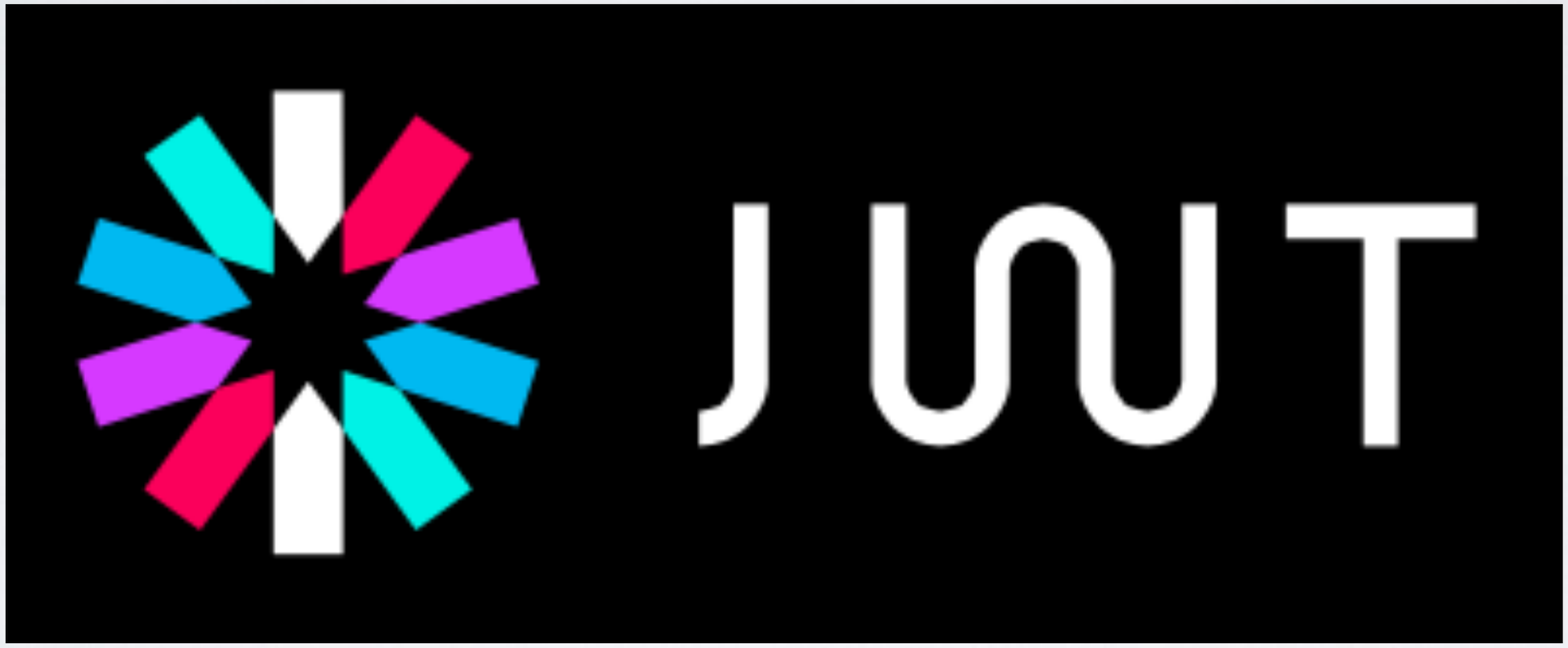


OAUTH AND SWAGGER DEMO



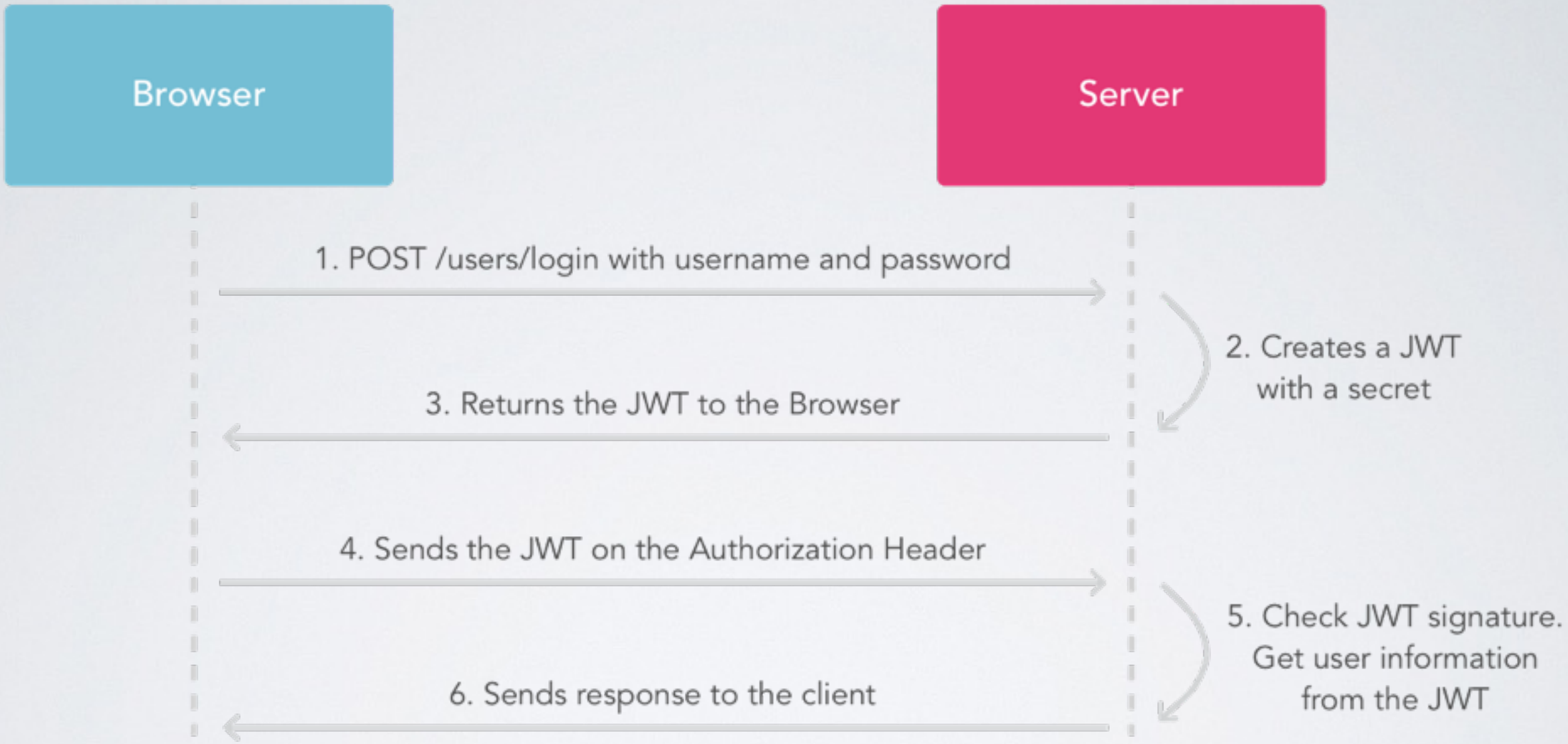
http://petstore.swagger.io/#/

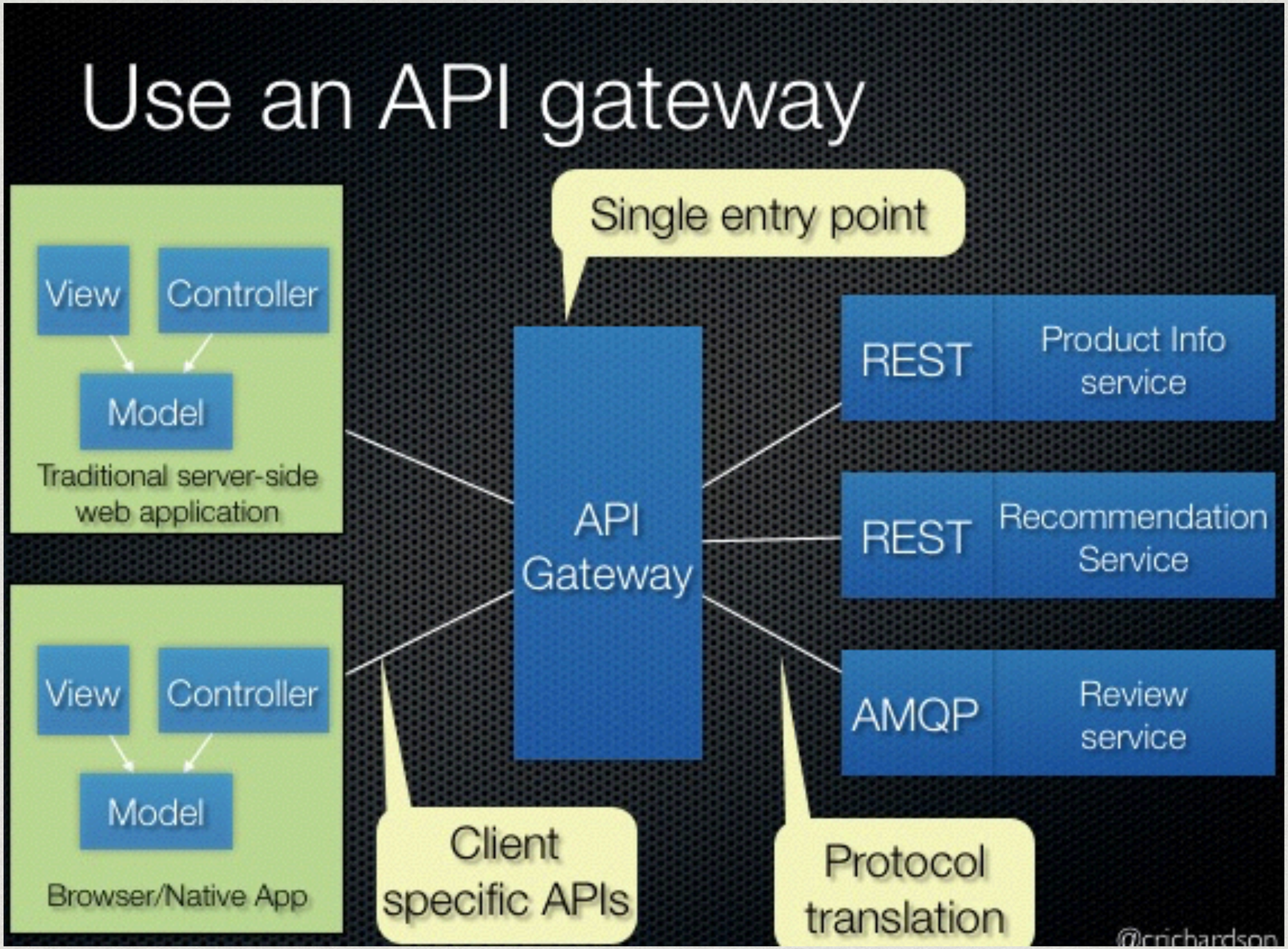
JSON WEB TOKENS



https://jwt.io/introduction/

Compact and self-contained way for securely transmitting information between parties as a JSON object


Browser
Server
1. POST /users/login with username and password
2. Creates a JWT
with a secret
3. Returns the JWT to the Browser
4. Sends the JWT on the Authorization Header
5. Check JWT signature.
Get user information
from the JWT
6. Sends response to the client



http://microservices.io/patterns/apigateway.html

API

```
<search>
    <items>
        <sku>555-ABC</sku>
        <quantity>1</quantity>
    </items>
</search>
```

HYPERMEDIA: LINKS IN API

```
<search>
    <items>
        <sku>555-ABC</sku>
        <quantity>1</quantity>
    </items>
    <id>156</id>
    <status>In Stock</status>
    <onHand>25</onHand>
    <links>
        <link rel="order" url="http://api.mycompany.com/orders/156" />
        <link rel="invoice" url="http://api.mycompany.com/payments/156" />
        <link rel="payment" url="http://api.mycompany.com/invoices/156" />
        <link rel="return" url="http://api.mycompany.com/returns/156" />
    </links>
</search>
```

JSON


```
{
  - _items: {
      - 1: {
            name: "Neil Young",
            created: "2013-04-29T13:47:59.918Z",
          - _links: {
              - self: {
                    href: "http://localhost:8080/api/artist/1"
                }
            },
      - 2: {
            name: "Joe Strummer",
            created: "2013-04-29T13:47:59.918Z",
          - _links: {
              - self: {
                    href: "http://localhost:8080/api/artist/2"
                }
            }
        }
    },
  - _links: {
      - parent: {
            href: "http://localhost:8080/api"
        }
    }
}
```

RETURNING MORE ACTIONS

```
{
   "Department":"Enforcement",
   "Id":"123",
   "Links":[
      {
         "Rel":"GetDetails",
         "Url":"/api/employees/56789"
      },
      {
         "Rel":"Fire",
         "Url":"/api/employees/56789"
      }
   ],
   "Name":"John Q Law"
}
```

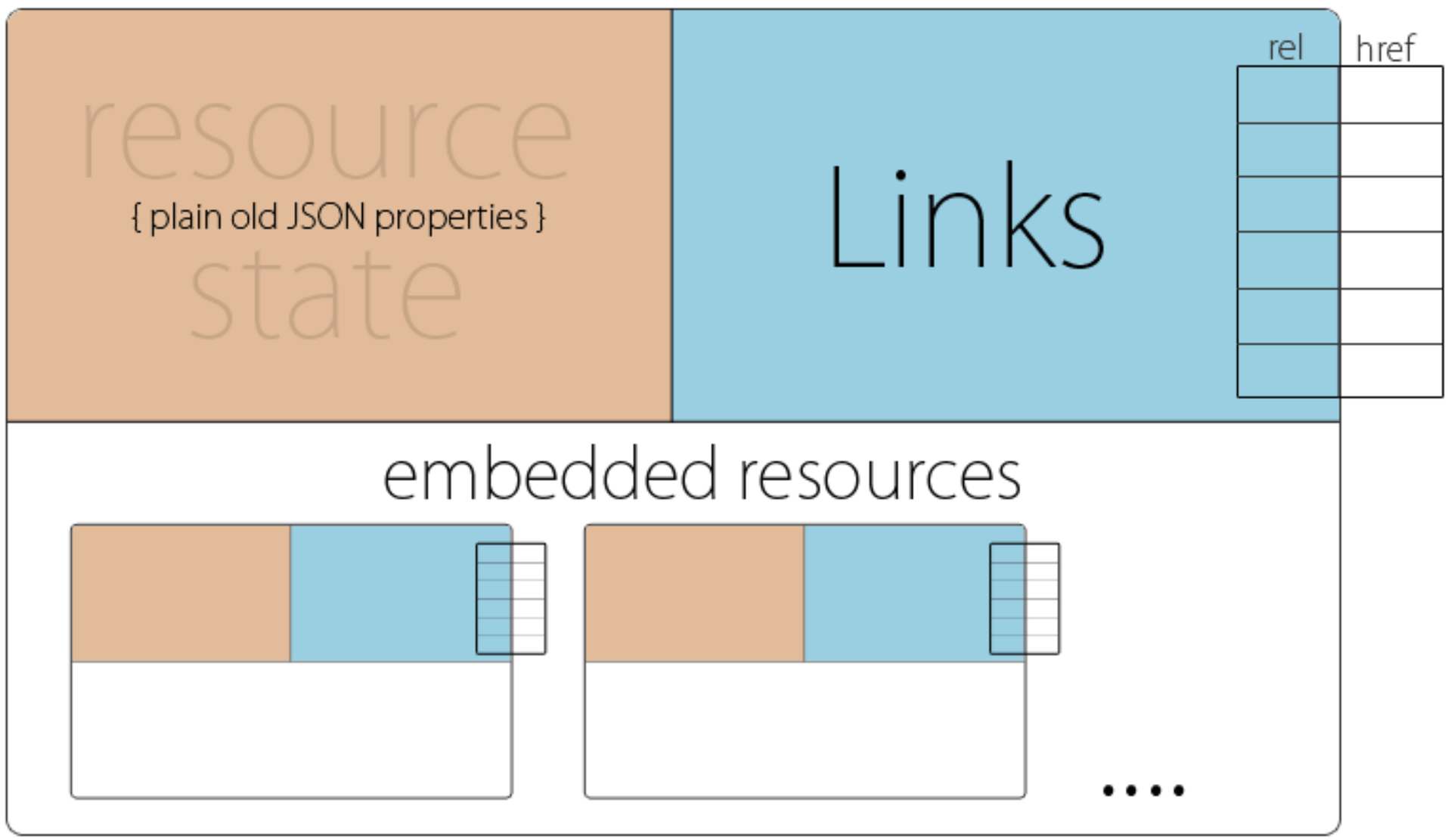
HAL: HYPERTEXT APPLICATION LANGUAGE

- HAL is a simple format that gives a consistent and easy way to hyperlink between resources in your API.

Amazon Chooses HAL Media Type for AppStream API

http://www.infoq.com/news/2014/03/amazon-hal-appstream

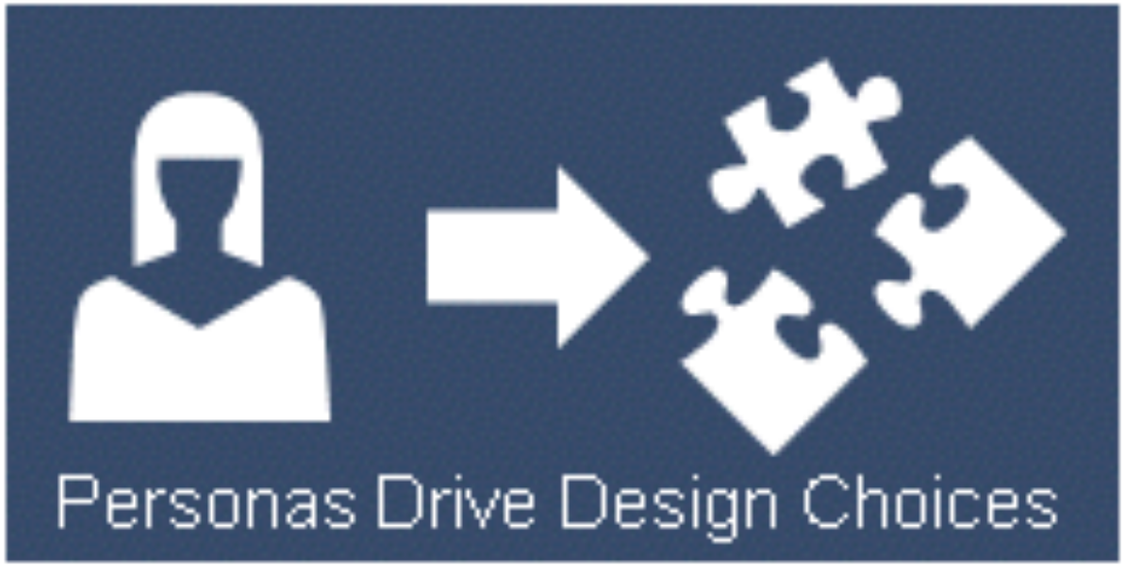
```
{
    "_links": {
        "self": { "href": "/orders" },
        "curies": [{ "name": "ea", "href": "http://example.com/docs/rels/{rel}", "templated": true }],
        "next": { "href": "/orders?page=2" },
        "ea:find": {
            "href": "/orders{?id}",
            "templated": true
        },
        "ea:admin": [{
            "href": "/admins/2",
            "title": "Fred"
        }, {
            "href": "/admins/5",
            "title": "Kate"
        }]
    },
    "currentlyProcessing": 14,
    "shippedToday": 20,
    "_embedded": {
        "ea:order": [{
            "_links": {
                "self": { "href": "/orders/123" },
                "ea:basket": { "href": "/baskets/98712" },
                "ea:customer": { "href": "/customers/7809" }
            },
            "total": 30.00,
            "currency": "USD",
            "status": "shipped"
        }, {
            "_links": {
                "self": { "href": "/orders/124" },
                "ea:basket": { "href": "/baskets/97213" },
                "ea:customer": { "href": "/customers/12369" }
            },
            "total": 20.00,
            "currency": "USD",
            "status": "processing"
        }]
    }
}
```

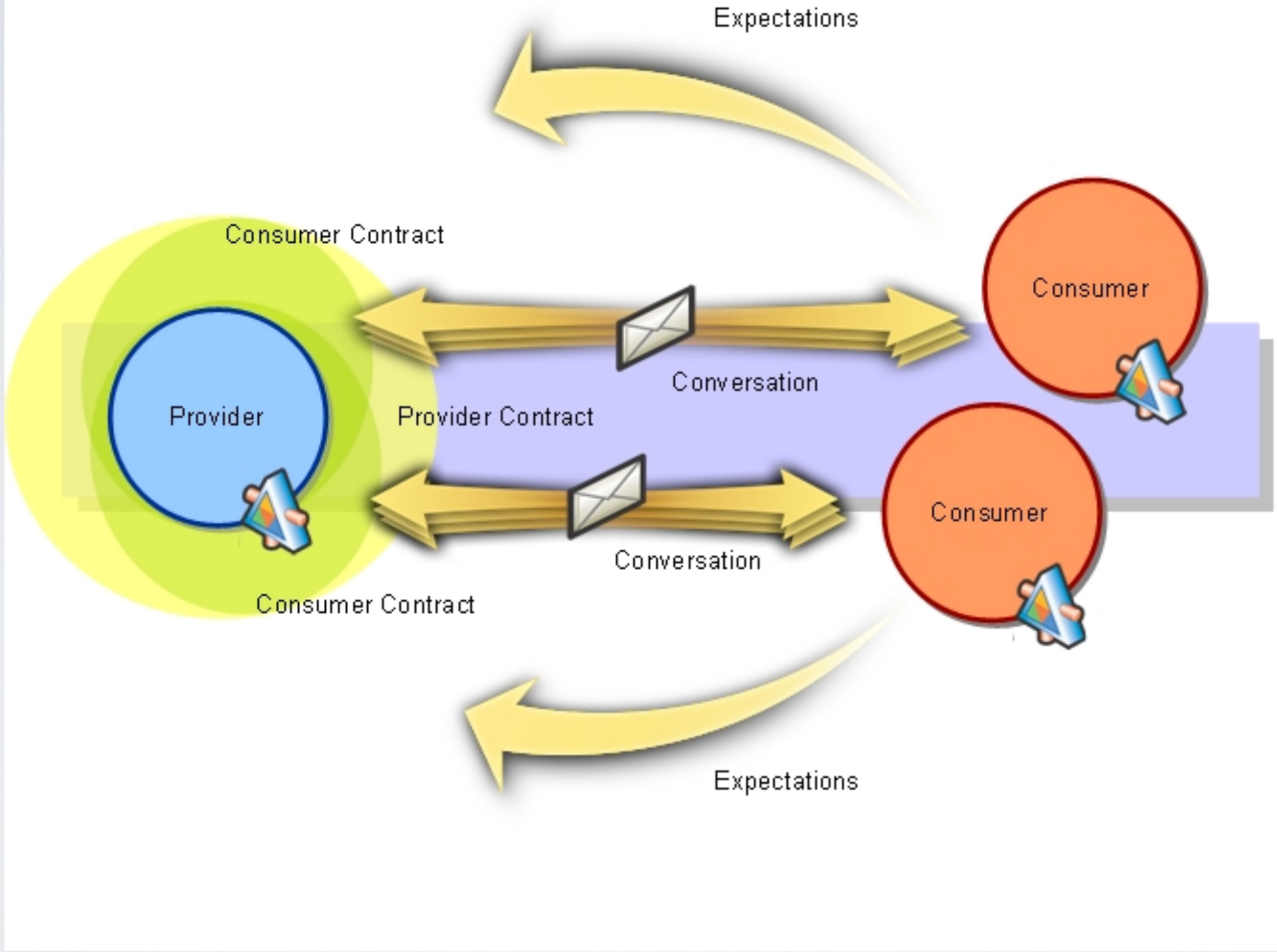


http://stateless.co/hal_specification.html

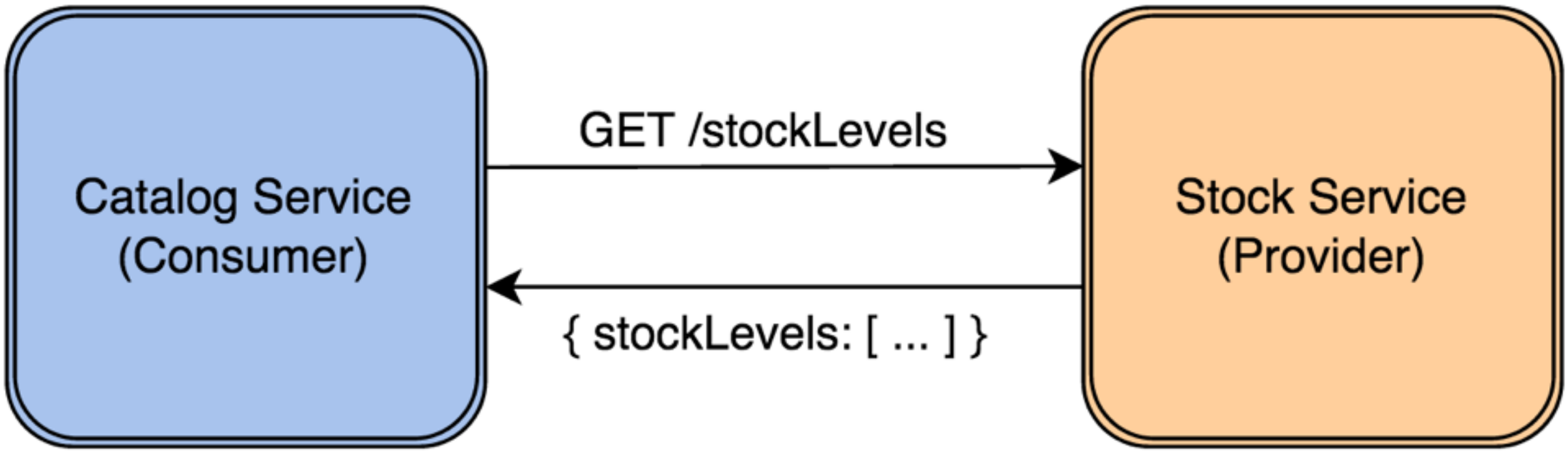
http://haltalk.herokuapp.com/explorer/browser.html#/

The Gartner Approach: Consumer-Driven API Design

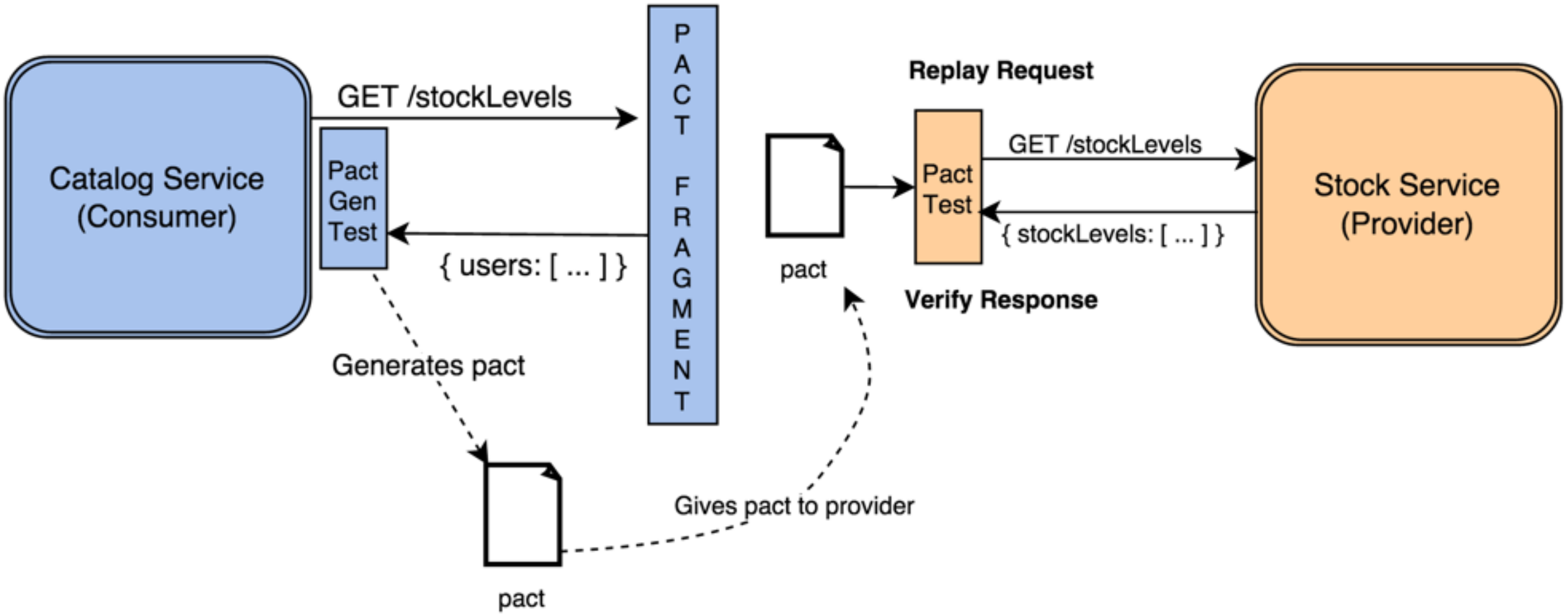




http://martinfowler.com/articles/consumerDrivenContracts.html



http://techblog.newsweaver.com/why-should-you-use-consumer-driven-contracts-for-microservices-integration-tests/


Catalog Service
(Consumer)
GET /stockLevels
Pact
Gen
Test
{ users: [...] }
PACT FRAGMENT
Generates pact
pact
Gives pact to provider
Replay Request
pact
Pact
Test
GET /stockLevels
{ stockLevels: [...] }
Verify Response
Stock Service
(Provider)

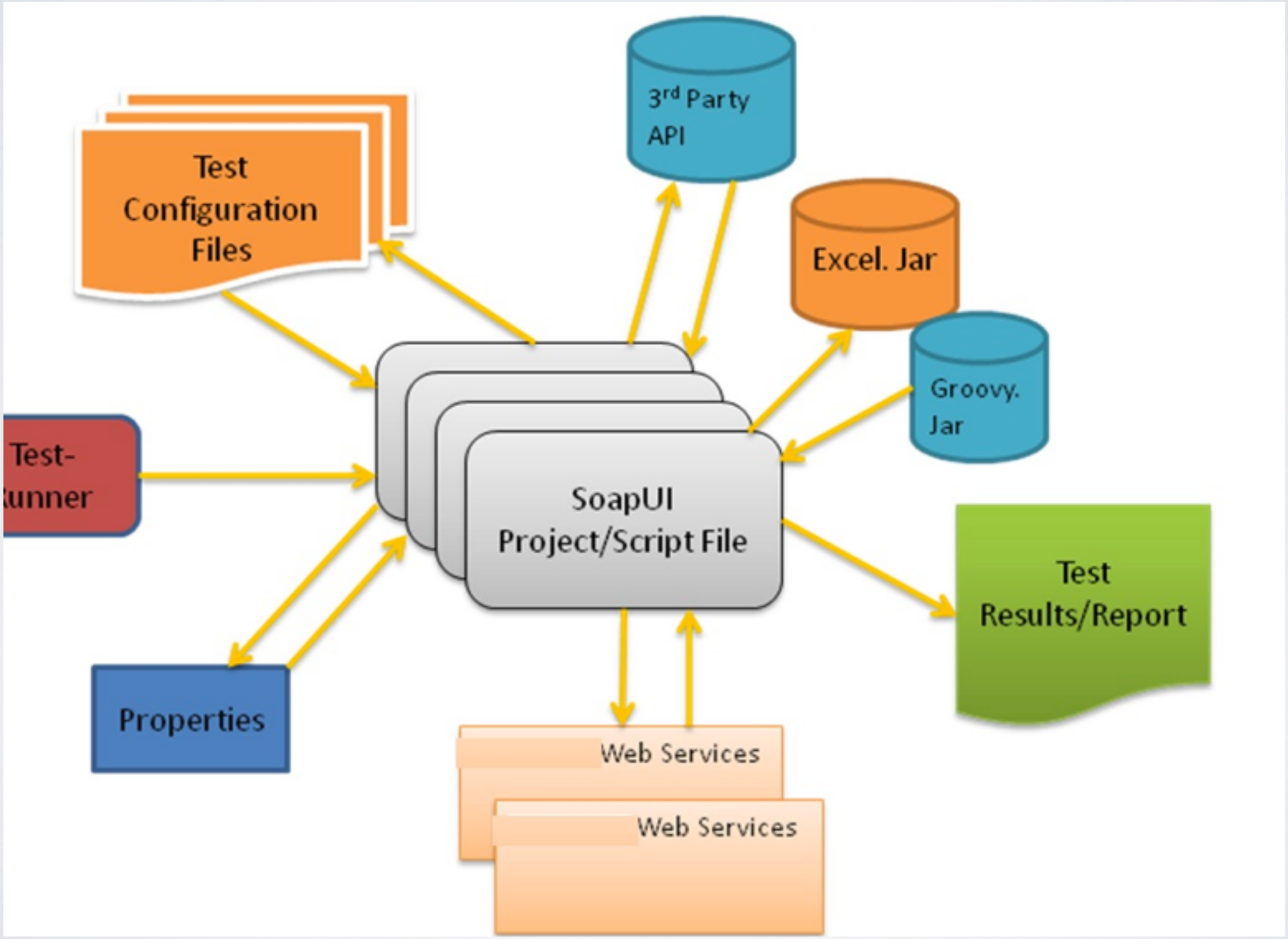
API MANAGEMENT

A Checklist for Every API Call

Managing the Complete API Lifecycle

apigee

TEST REST APIS



POSTMAN DEMO

DESIGNING RESTFUL APIS

- RESTful APIs
- Hypermedia
- Principles for designing APIs
- SAML and OAuth
- Customer driven contract
- API testing