

A composite image featuring two men from the TV show MythBusters. The man on the left, wearing glasses and a dark shirt, is smiling. The man on the right, wearing a beret and a light-colored shirt, has a more serious expression. They are standing in front of a blue background with white stars and some faint, illegible text. A large, stylized yellow title 'HYPERMEDIA' is overlaid at the top. Below them, a white banner with the words 'MYTHBUSTERS' in a large, orange, flame-like font is held across the bottom.

# HYPERMEDIA

## MYTHBUSTERS

Glenn Block, Director of Product Management, Auth0



# Auth0

Identity made simple for developers

<https://auth0.com>

# My love affair with hypermedia started in 2010

Taking HTTP support in WCF to the next level

The Real Glenn Block October 25, 2010

Rate this article ★★★★☆

f 0    t 0    ln 0    c 0

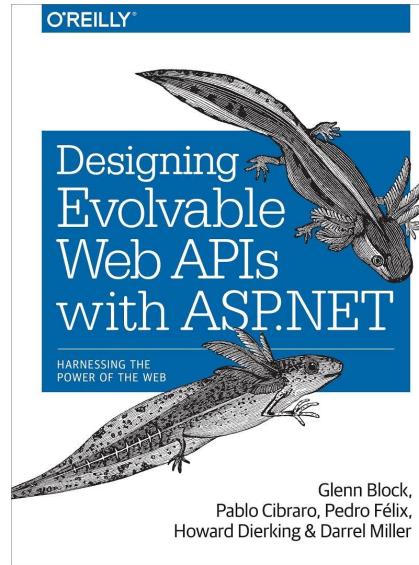


If you are building for the web and the cloud, then you know that HTTP is important. WCF is no stranger to HTTP. Since .NET 3.5, we've been continually delivering capabilities in the platform to enable developers to expose services as resources over HTTP. Now we're looking to raise the bar several notches.

At PDC, I'll be showing work we are doing to unlock the full richness of HTTP within your network based applications.

- We're going to let you have complete control over HTTP.
- Support a multitude of media types (formats) i.e XML, JSON, ATOM, OData, and custom formats including those that are hypermedia driven.
- Give you full access to your Uri and headers,
- Provide richer support for web frameworks like JQuery
- Provide a more simplified configuration story.

That is just the beginning. We want you to have it your way with HTTP!



## CollectionJson.NET

MyGet Successful

This library provides support for using the [Collection+JSON](#) hypermedia mediatype authored by [Mike Amundsen](#).



# CHAPTER 5

## Representational State Transfer (REST)

This chapter introduces and elaborates the Representational State Transfer (REST) architectural style for distributed hypermedia systems, describing the software engineering principles guiding REST and the interaction constraints chosen to retain those principles, while contrasting them to the constraints of other architectural styles. REST is a hybrid style derived from several of the network-based architectural styles described in Chapter 3 and combined with additional constraints that define a uniform connector interface. The software architecture framework of Chapter 1 is used to define the architectural elements of REST and examine sample process, connector, and data views of prototypical architectures.

### 5.1 Deriving REST

The design rationale behind the Web architecture can be described by an architectural style consisting of the set of constraints applied to elements within the architecture. By examining the impact of each constraint as it is added to the evolving style, we can identify the properties induced by the Web's constraints. Additional constraints can then be applied to form a new architectural style that better reflects the desired properties of a modern Web architecture. This section provides a general overview of REST by walking through the process of deriving it as an architectural style. Later sections will describe in more detail the specific constraints that compose the REST style.

***“This chapter introduces and elaborates the Representational State Transfer (REST) architectural style for distributed hypermedia systems”***



[Home](#) [About](#) [Archives](#) [Links](#)

# Untangled

*musings of Roy T. Fielding*

PHOTO BY PATRICK LIENZ

Mon  
20 Oct  
2008

## REST APIs must be hypertext-driven

Posted by Roy T. Fielding under software architecture, web architecture

[51] Comments

I am getting frustrated by the number of people calling any HTTP-based interface a REST API. Today's example is the [SocialSite REST API](#). That is RPC. It screams RPC. There is so much coupling on display that it should be given an X rating.

What needs to be done to make the REST architectural style clear on the notion that hypertext is a constraint? In other words, if the engine of application state (and hence the API) is not being driven by hypertext, then it cannot be RESTful and cannot be a REST API. Period. Is there some broken manual somewhere that needs to be fixed?

### Archived Entry

**Post Date :**  
Monday, Oct 20th, 2008  
at 5:20 am

**Category :**  
software architecture  
and web architecture

**Tags :**  
hypertext, REST

**Do More :**  
Both comments and  
pings are currently  
closed.

Search

### Categories

[blogging \(4\)](#)

***“In other words, if the engine of application state (and hence the API) is not being driven by hypertext then it cannot be RESTful and cannot be a REST API”***

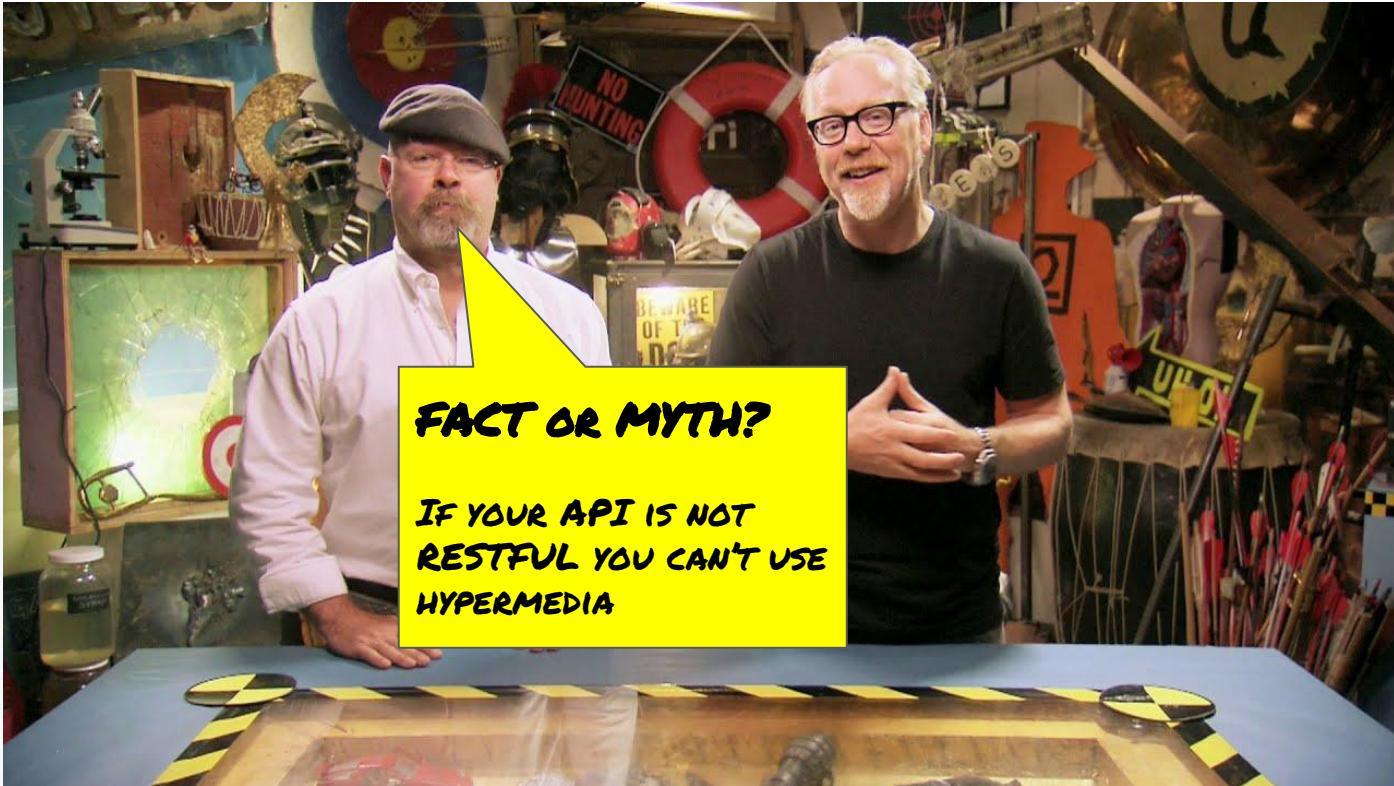


Auth0

<http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven>

# CONFIRMED

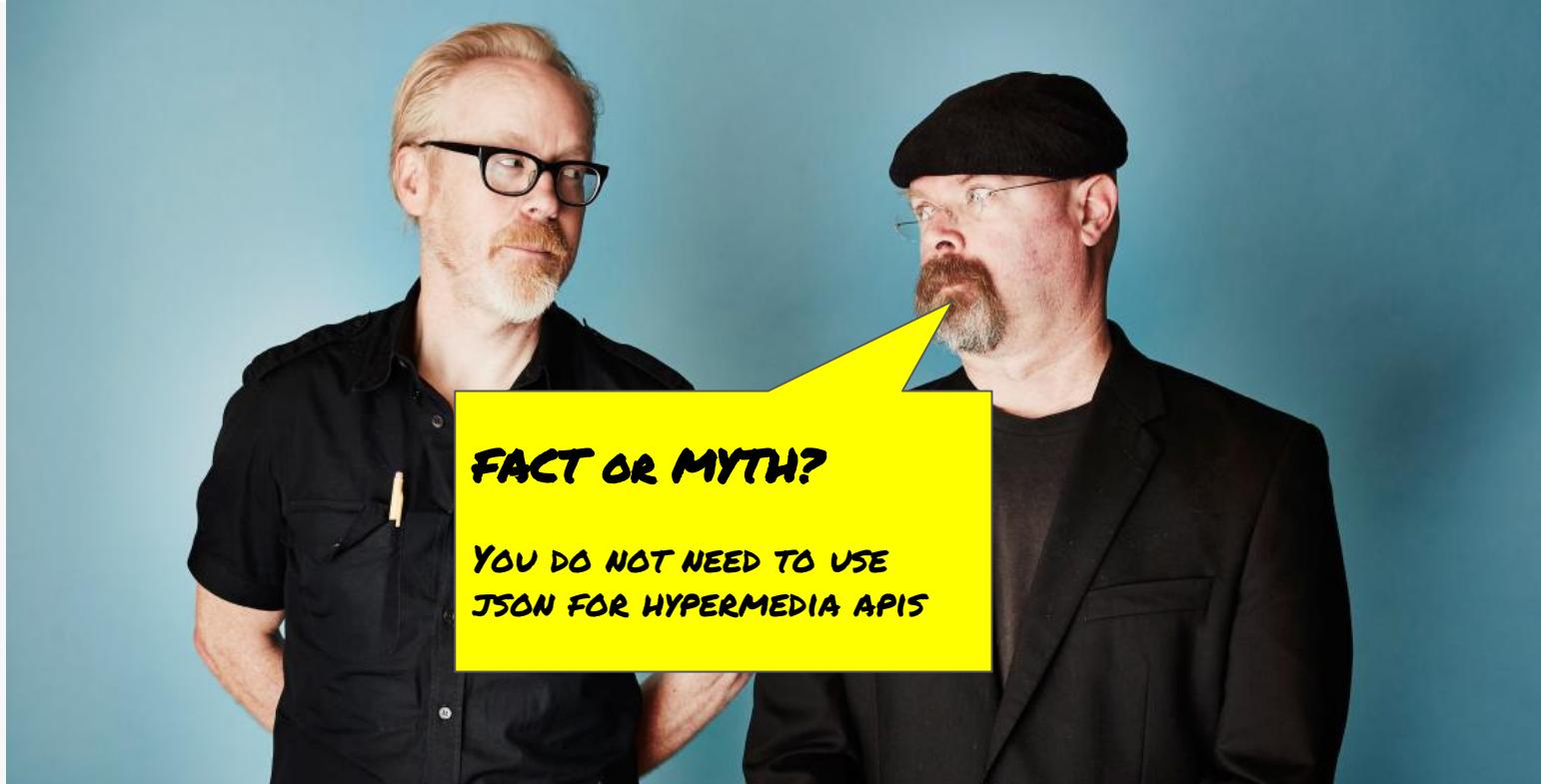




POST http://rpcalldaylong.com/rpcapi/dosomething

```
{  
  "response": "I am as RPC as you can get",  
  "links": {  
    "doSometing": "...",  
    "doSomethingElse": "...",  
  }  
}
```

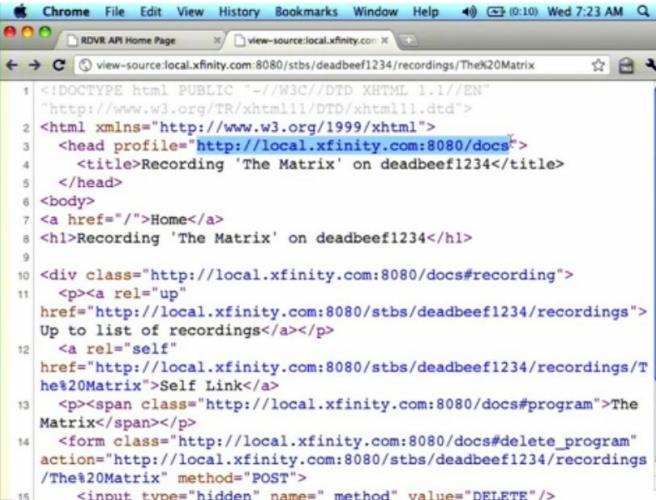




# Building Hypermedia APIs with HTML



Jon Moore (@jon\_moore)  
Technical Fellow, Chief Engineer

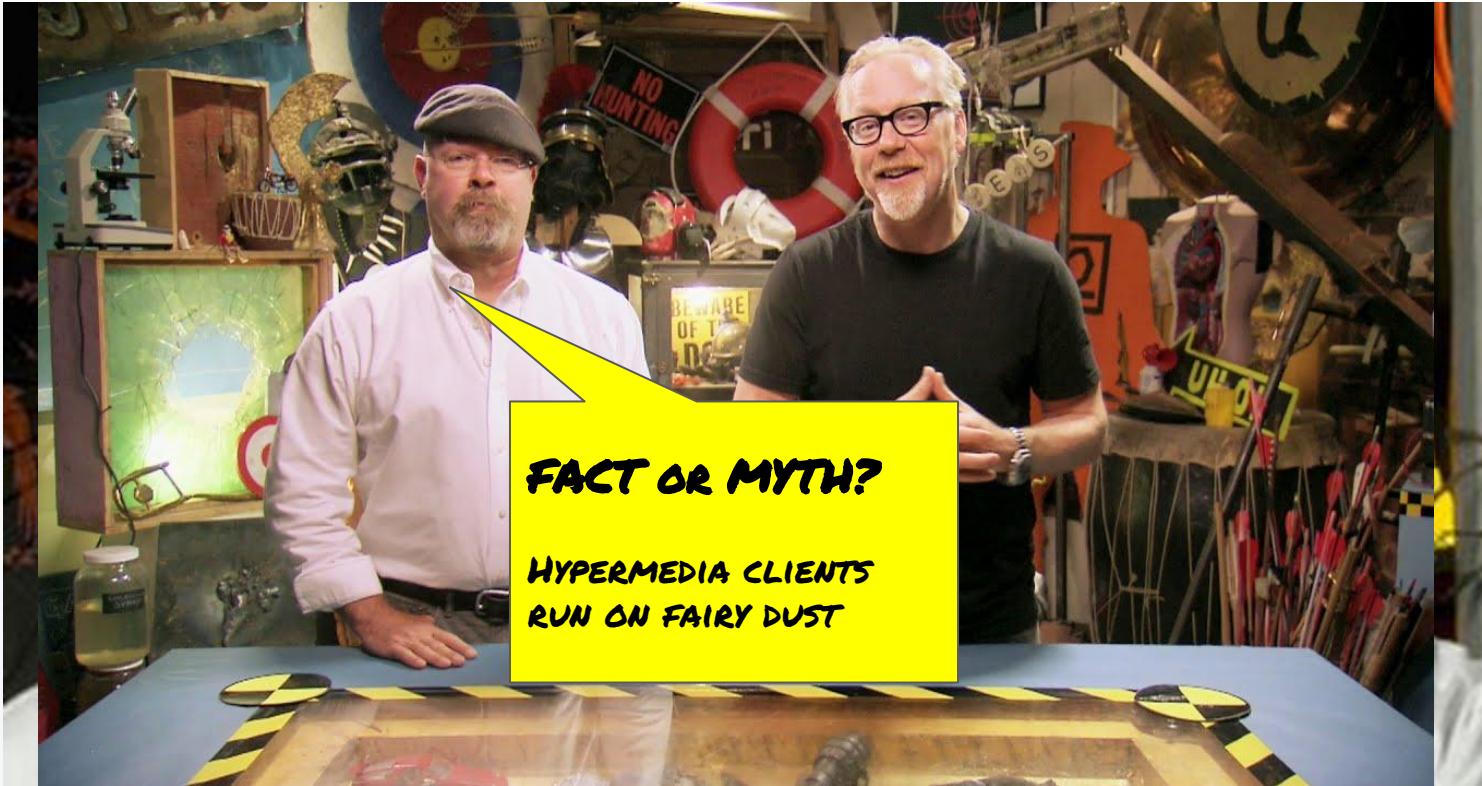


The screenshot shows a Chrome browser window with the title "RDVR API Home Page". The URL in the address bar is "view-source:local.xfinity.com:8080/stbs/deadbeef1234/recordings/The%20Matrix". The page content is the HTML source code for a recording page. The code includes a DOCTYPE declaration, XML namespace declarations, and various HTML elements like , , , 

# , , , and . The code uses class and href attributes to define hyperlinks and styles. Conclusions has features for - Domain Semantics - Workflow - Request Recipes HTML APIs **accessible** to - Programs - Developers - Other humans Web accessible to - Humans - Programs?



**CONFIRMED**





# Traverson

A Hypermedia API/HATEOAS Client for Node.js and the Browser

build passing



File Size (browser build)	KB
minified & gzipped	13
minified	44



# JSON-LD

JSON-LD is a lightweight Linked Data format. It is easy for humans to read and write. It is based on the already successful JSON format and provides a way to help JSON data interoperate at Web-scale. JSON-LD is an ideal data format for programming environments, REST Web services, and unstructured databases such as CouchDB and MongoDB.





# Why I Hate HATEOAS

Most of the population of people who have read [Roy Fielding's dissertation](#) tell us that, while we may understand HTTP verbs, we don't know anything about *hypermedia*. Our REST APIs, they say, are not *really* RESTful. They certainly do not exhibit Hypertext As The Engine Of Application State, or **HATEOAS**. The fact that we have to need to read documentation to understand what action to take next using only the contents of the HTTP response.

## Why

When does  
you're ask  
you come  
you if the

When he a  
necessary  
explicit do

<https://je>

## Platform Strategy & Architecture

HOME SAM PATTERN ABOUT CONTACT

Home » APIs » Hypermedia: don't bother



## Hypermedia: don't bother

Jj

April 30, 2015

1 Comment

<http://www.ebpml.org/blog15/2015/04/hypermedia-dont-bother/>

# Getting hyper about hypermedia APIs



David wrote this on Dec 20 2012 / [56 comments](#)

[Tweet](#)

[Share on Facebook](#)

[Share on LinkedIn](#)

Using URLs instead of ID references in your APIs is a nice idea. You should do that. It makes it marginally more convenient when writing a client wrapper because you don't have to embed URL templates. So you can do

Jeremiah Lee  
[JeremiahLee](#)

[Follow](#)

`client.get("/people/#`

he I hope the conversation about media makes it beyond #APICraft to the us because it still seems like 99% bullshit.

LIKES

6



3 Jul 2014

cisco

6

...

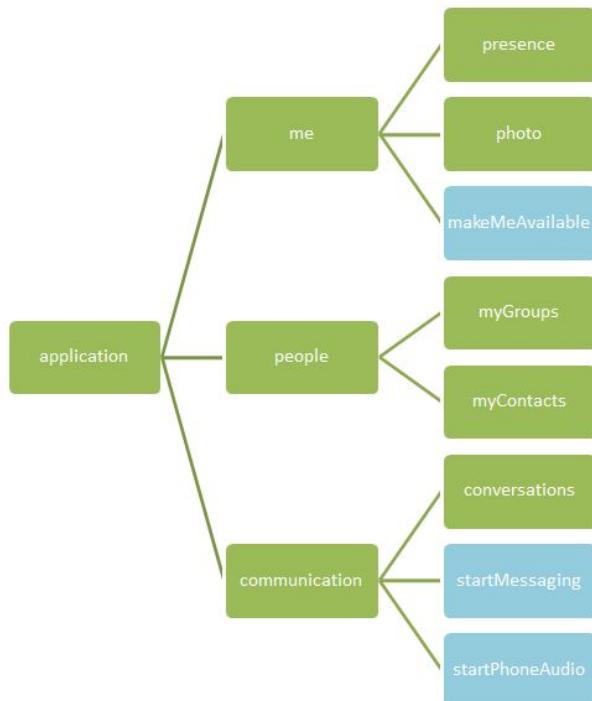
is completely overblown.  
ard you from breakage, it's  
andardizing API clients, and

r-about-hypermedia-apis

<https://twitter.com/JeremiahLee/status/493830026983247876>

# Microsoft's Unified Communication Web API (UCWA)

## Value - Discoverability



The Microsoft Unified Communications Web API 2.0 uses hypermedia links to discover and dynamically navigate the features that are available at any given time.

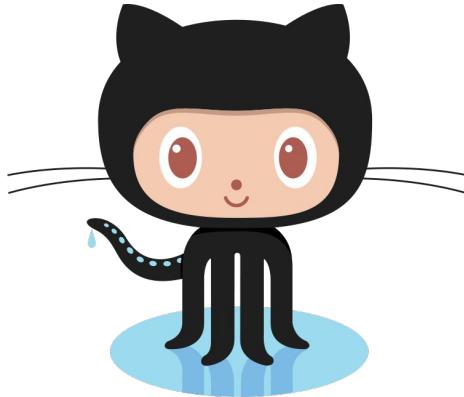
The client application starts with application creation and is presented with the features that are available in the returned **application** resource.

Some links are temporary and reflect the capabilities that the user has at a particular moment in time. For example, when a meeting attendee is in the lobby, an admit capability will appear. After the user is admitted, the admit link will no longer appear.

# Github API

## Value - Version tolerance

### Hypermedia



All resources may have one or more `*_url` properties linking to other resources. These are meant to provide explicit URLs so that proper API clients don't need to construct URLs on their own. It is highly recommended that API clients use these. Doing so will make future upgrades of the API easier for developers. All URLs are expected to be proper [RFC 6570](#) URI templates.

You can then expand these templates using something like the [uri\\_template](#) gem:

```
>> tmpl = URITemplate.new('/notifications{?since,all,participating}')
>> tmpl.expand
=> "/notifications"

>> tmpl.expand :all => 1
=> "/notifications?all=1"

>> tmpl.expand :all => 1, :participating => 1
=> "/notifications?all=1&participating=1"
```

# Guardian API

## Value - Discoverability / Evolvability / Agility

### Discoverable

**API federation**  
Links to Media API from our Content API

Service index in Media API

```
{  
  - data: {  
    description: "This is the Media API"  
  },  
  - links: [  
    - {  
      rel: "search",  
      href: "https://api.media.gu.com/images/?q,ids,offset,length,fromDate,toDate,orderBy,s"  
    },  
    - {  
      rel: "image",  
      href: "https://api.media.gu.com/images/{id}"  
    },  
    - {  
      rel: "cropper",  
      href: "https://cropper.media.gu.com"  
    },  
    - {  
      rel: "loader",  
      href: "https://loader.media.gu.com"  
    },  
  ]  
},
```



```
{  
  type: "image",  
  title: "http://media.guim.co.uk/a1084e2d8d1f5a63aab2fa610ee618308987df5c",  
  typeData: {  
    mediaId: "a1084e2d8d1f5a63aab2fa610ee618308987df5c",  
    mediaApiUri: "https://api.media.gu.com/images/a1084e2d8d1f5a63aab2fa610ee618308987df5c",  
    caption: "A group of people look at the body of",  
    credit: "Photograph: Jeff Chiu/AP",  
    width: "2000",  
    height: "1200"  
  },  
},
```

### Evolveable

1 Persistent URIs  
3 Clients rely on links, not out-of-band config

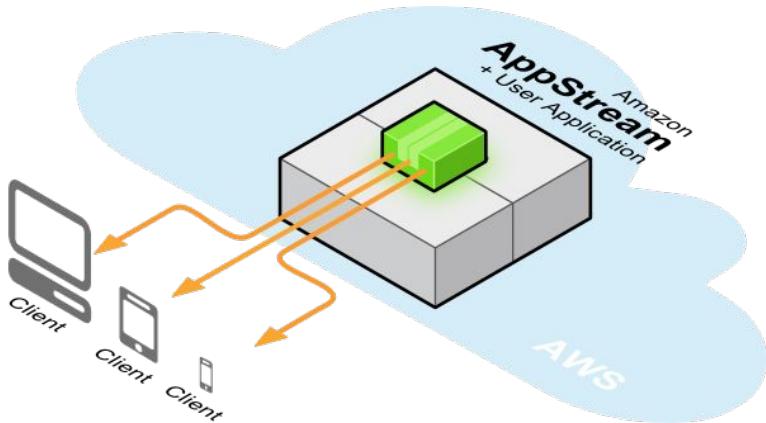
<b>Server configures clients</b> Only config is API root No URL construction Change link targets transparently	<b>Extensible</b> Add data properties Add links & form fields Ignored if unsupported
---	---

### Composeable

Add new services via links and forms  
Shared media-type helps  
Avoids monolithic library

# Amazon App Streaming API

## Value - Discoverability



### Link Relations

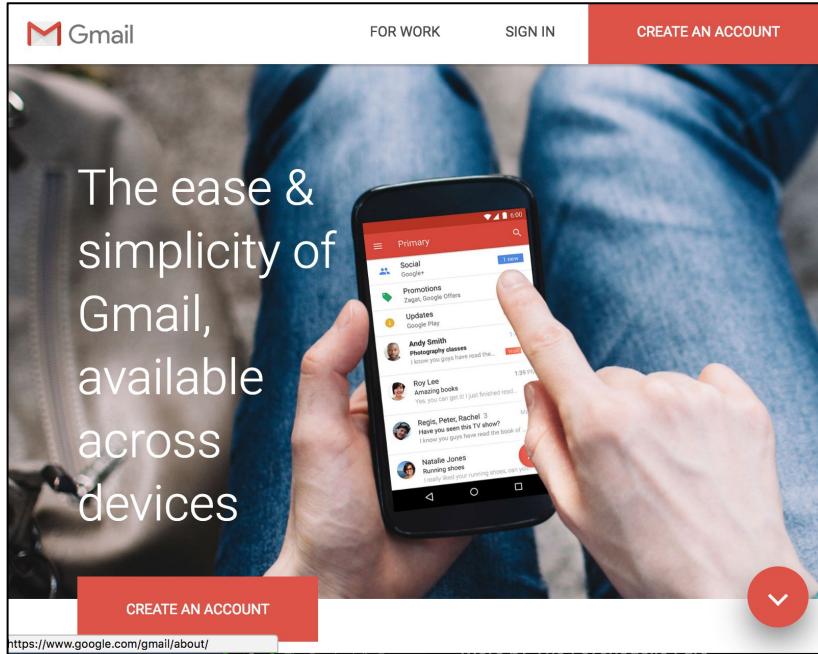
The Amazon AppStream API provides the following link relations that you can use to access and modify Amazon AppStream resources.

#### Topics

- [appstream:applications](#)
- [application:by-id](#)
- [application:create](#)
- [application:update](#)
- [application:archive](#)
- [application:reactivate](#)
- [application:delete](#)
- [application:status](#)
- [application:errors](#)
- [application:sessions](#)
- [session:by-id](#)
- [session:entitle](#)
- [session:status](#)
- [session:terminate](#)
- [Common Link Relations](#)

# Gmail

## Value - Discoverability



The ease & simplicity of Gmail, available across devices

FOR WORK SIGN IN CREATE AN ACCOUNT

CREATE AN ACCOUNT

<https://www.google.com/gmail/about/>

### JSON-LD



JSON-LD is an easy-to-use JSON-based linked data format that defines the concept of `context` to specify the vocabulary for types and properties. Gmail supports [JSON-LD data embedded in HTML documents](#) with the `@context` of [schema.org](#), as in the following example:

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Person",
  "name": "John Doe",
  "jobTitle": "Graduate research assistant",
  "affiliation": "University of Dreams",
  "additionalName": "Johnny",
  "url": "http://www.example.com",
  "address": {
    "@type": "PostalAddress",
    "streetAddress": "1234 Peach Drive",
    "addressLocality": "Wonderland",
    "addressRegion": "Georgia"
  }
}</script>
```

★ Note: the `@type` key is a reserved key name and should contain either a full type URI or a URI fragment (in which case a <http://schema.org> prefix, derived from the supplied data-context attribute, is assumed).

The full specifications and requirements for the JSON-LD syntax are available on [json-ld.org](#), and you can also use our [Schema Validator](#) tool to try out JSON-LD and debug your markup.

# Zetta

## Value - Discoverability / Device management

The screenshot shows the Zetta website homepage. At the top, there's a navigation bar with links for 'PROJECTS', 'COMMUNITY', 'DOCS', 'MODULES', and 'GITHUB'. The main title 'An API-First Internet of Things Platform' is prominently displayed. Below it, a paragraph explains that Zetta is an open source platform built on Node.js for creating Internet of Things servers. It also highlights that Zetta combines REST APIs, WebSockets, and reactive programming, making it perfect for real-time applications. At the bottom left, there's a link to 'http://zettajs.org'.

zetta

PROJECTS COMMUNITY DOCS MODULES GITHUB

## An API-First Internet of Things Platform

Zetta is an open source platform built on **Node.js** for creating Internet of Things servers that run across geo-distributed computers and the cloud. Zetta combines **REST APIs**, **WebSockets** and **reactive programming** – perfect for assembling many devices into data-intensive, real-time applications.

---

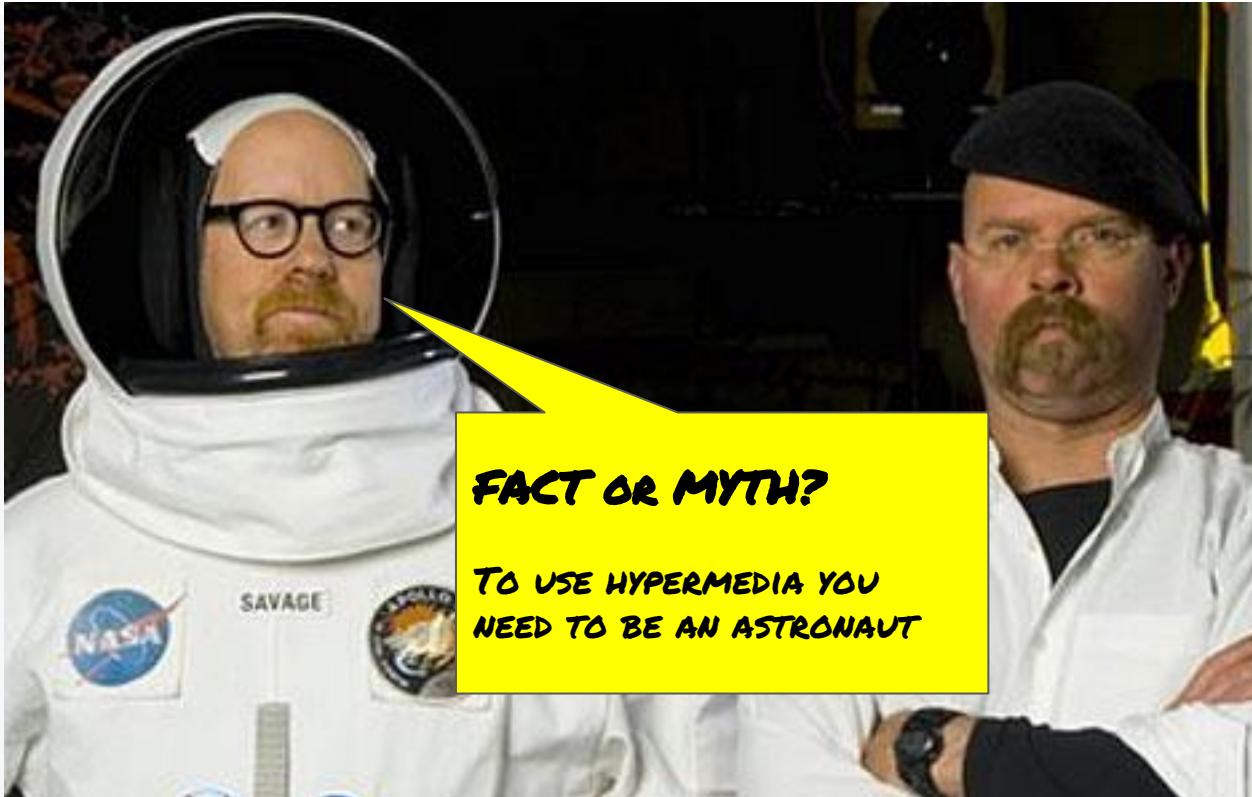
<http://zettajs.org>

```
GET /servers/321/devices/123
Host: cloud.herokuapp.com
Accept: application/vnd.siren+json

{
  "class": [
    "device"
  ],
  "properties": {
    "id": "123",
    "type": "arm",
    "name": "Robot Arm",
    "state": "standby"
  },
  "actions": [
    {
      "name": "move-claw",
      "method": "POST"
    }
  ]
}
```

# CONFIRMED









**FACT OR MYTH?**

**IF YOU ARE NOT USING  
HYPERMEDIA YOUR API IS  
USELESS**



# Takeaways

**Hypermedia adoption is rising**

**Hypermedia does provide business value**

**You do not have to build a hypermedia API**

## More information on hypermedia

Mike Amundsen's books - <http://www.oreilly.com/pub/au/1192>

Steve Klabnik - <https://www.infoq.com/articles/implementing-hypermedia>

Darrel Miller - <http://www.bizcoder.com/>

JSON-LD - <http://json-ld.org/>

A photograph of two men from the TV show MythBusters. The man on the left is wearing glasses and a dark shirt, smiling. The man on the right is wearing a beret and a white lab coat, looking slightly upwards. They are standing in front of a blue background with some faint text and small lights. Large yellow text overlaid on the image reads "THANK YOU!" and below it, a banner with the words "MYTHBUSTERS" in a white, outlined font, set against a background that looks like it's on fire.

Glenn Block, Director of Product Management, Auth0