Enterprise API Management

RESTful Web API Design

Steffen Fohn, Ph.D.
Chief Data Architect, ADP
June 4, 2019 | Rockville, MD
# Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADP Overview</td>
<td>03</td>
</tr>
<tr>
<td>ADP Marketplace</td>
<td>04</td>
</tr>
<tr>
<td>ADP Integration</td>
<td>05</td>
</tr>
<tr>
<td>What is a RESTful Web API?</td>
<td>06</td>
</tr>
<tr>
<td>What is a “Standard” RESTful Web API?</td>
<td>09</td>
</tr>
<tr>
<td>An Operational Model for RESTful Web API Design</td>
<td>17</td>
</tr>
</tbody>
</table>
ADP Overview

Comprehensive global provider of HCM solutions
uniting HR, payroll, talent, time, tax and benefits,
a leader in business outsourcing services, analytics and compliance expertise

Nearly 70 years of history
$13+ B Revenue
58000 Associates
175+ Countries
740,000+ Clients
Pay 1 in 6 Workers
~ $2 trillion in Annual Money Movement
ADP Marketplace

What are you looking to improve?

You may only need easy-to-use payroll apps, or you may be looking for other ways to make other aspects of HR simpler. Select a solution to get started.

A digital storefront offering a collection of solutions built by ADP and its Marketplace Partners ready to simply and securely share data with your ADP platform.

- Recruiting & Onboarding
  Find and hire the right people.

- Benefits Administration
  Offer employees a full benefits experience.

- Performance
  Help your business and employees achieve more.

- Data Connectors
  Connect your existing HR solutions with your ADP platform.

- Travel & Expense
  Simplify expense management.

- Compensation
  Understand, benchmark and communicate employee pay.

- Time & Labor Management
  Better plan, track and schedule workers time.

- Financial Wellness
  Help your workforce build financial confidence.

- Productivity & Collaboration
  Simplify how work gets done.

- Benefits
  Provide the benefits your employees deserve.

- Learning Management
  Develop and train your employees.

- Point of Sale Connectors
  Connect your POS software for payroll integration.

- Reward & Recognition
  Say thank you for a job well done.

- Compliance & Consulting
  Help to minimize risk with confidence.
ADP Integration (on the outside and the inside)

Our standard RESTful Web APIs are a Key Enabler for External Integration with ADP Products in the Marketplace as well as Internal Integration across ADP’s Products.

• Some of our RESTful Web API History
  ◦ Introduced as part of ADP Mobile Solutions
    - The ADP Mobile app was first released in 2011
    - Today, 20 million registered uses, performing more than 1,000 HR transaction per second

• RESTful Web API Design Standard
  ◦ Ensured consistent API designs across ADP products
  ◦ The first version of the design standard was completed in 2010.
  ◦ The most recent version of our design standard, in support of the ADP overall integrations, was completed in April, 2019.
  ◦ In 2018 ADP contributed its RESTful Web API Design standard to OAGi.
What is a RESTful Web API anyway?

Some essential terms and definitions...

- **A Service** - A software program that makes its functionality available via a published API. [Erl et al. (2012)]

- **An API** - An Application Programming Interface (API) is a named set of operations (functions and data) that is offered by a service provider (i.e., Server) and used by service consumer (i.e., Client) to enable communication between the components. [OMG’s UML]

- **A Web Service** – A Service exposed on the World Wide Web

- **A Web API** – An API offered by a Web Service

- **REST** - Representational State Transfer or Representational Entity State Transfer
  ◦ Architectural style, named and defined by Roy Fielding in 2000 in his Ph.D. dissertation that describes the Web’s architectural style

- **RESTful Web API** - A Web API that conforms to the **REST architectural style**.

**RESTful Web API = (((Service + API) + Web) + REST)**
What is a RESTful Web API anyway?

Fielding's description of the REST architectural style consisted of constraints in six categories ... (rules that give the WWW its desirable architectural properties) ...

• Client-Server – separation of concerns
• Stateless - session state is kept on the client
• Cacheable – data is labelled as cacheable or non-cacheable
• **Uniform Interface** - “The central feature that distinguishes REST from other network-based styles is its emphasis on uniform interface between components”.
  ◦ “REST is defined by four interface constraints:
    - identification of resources;
    - manipulation of resources through representations;
    - self-descriptive messages; and,
    - hypermedia as the engine of application state.” [Fielding (2000)]
• Layered System – allows for additional mediator components (e.g., proxies, gateways) to add features to the system.
• Code-On Demand – client functionality extended through download of code or script

**RESTful Web API = (((Service + API) + Web) + REST)**
What is a RESTful Web API anyway?

Understand what it means for a Web API to be “RESTful”, then enforce that understanding.

Richardson’s Maturity Model for RESTful Web APIs ... consists of four levels; each level declares an aspect of the Web’s Uniform Interface that must be satisfied for a given maturity level.

• Level 0
  ◦ Must use the HTTP protocol for communication transport

• Level 1
  ◦ Meets Level 0 requirements
  ◦ Must use Resources

• Level 2
  ◦ Meets Level 1 requirements
  ◦ Must use HTTP Verbs and HTTP Response Codes

• Level 3
  ◦ Meets Level 2 requirements
  ◦ Must use Hypermedia controls (Hypertext As The Engine of Application State, HATEOAS)

[Richardson (2008)]

HATEOAS is a constraint to being RESTful. [Fielding (2000)]
What is a **Standard RESTful Web API**?

Understand the myriad of API-related design concerns, then address and enforce in a common approach. Some of those concerns include:

- Approach to API Versioning
- Adoption of HTTP Headers
- Conventions for URI Design and Format
- Types of Resources managed in the APIs
- Approach to Managing Resources (Create, Read, Update, Delete)
  e.g. how to filter?, how to select?, how to pagination?
- Adoption of HTTP Response Statuses

- Approach for a standard response message to communicate successes, warnings, failures
- Support of Hypermedia Controls
- Accommodations for large URIs and query components with sensitive data
- Interaction patterns for asynchronous communication
- Conventions for Message Body Representation (metadata and resource representation)
A “Design Standard” for RESTful Web APIs, addresses this “myriad of other API-related design concerns” in a manner that realizes a level 3 maturity. It...

- Describes the techniques, patterns and formats to facilitate consistent, efficient design and use of Web APIs.
- Provides a common basis upon which RESTful Web API Specifications are developed and maintained.

Note: A given RESTful Web API Specification, designed to meet certain integration requirements, need not address the full breadth or scope of design considerations addressed in the standard i.e. leverage a subset of the standard

- Should be independent of any specific resource representation format (e.g. JSON, XML).
  - For example, a resource representation-specific Naming and Design Rules (NDR) are not addressed in this standard.

**Standard RESTful Web API = (((Service + API) + Web) + REST + Design Standard)**
What is a **Standard RESTful Web API**?

The target audience of a "REST Standard" for Web APIs is ...

- The API Designer and Implementer
  - To guide and support API Specification and Implementation, respectively, of services interfaces exposed on the web according to the REST architectural style

**Standard RESTful Web API = ((((Service + API) + Web) + REST + Design Standard)**
A RESTful Web API Design Standard

OAGi works on the design standard... in the Mobile, Cloud, and JSON Working Group.
A RESTful Web API Design Standard

Section 4.0, Versioning for the API Specification

Leverages Semantic Versioning

- Major Version - indicates a new version of the artifact that breaks backwards compatibility
- Minor Version - indicates a new version of the artifact that maintains backwards compatibility.
- Patch Version - indicates a bug fix that resolves incorrect behavior.

[Preston-Werner, https://semver.org]
A RESTful Web API Design Standard

Section 4.0, Versioning for the API Specification

For any artifact (Web API or constituent part), the following rules apply:

R5  Version and revision information MUST be expressed in the form
    majorVersionID.minorVersionID.patchVersionID

R6  Use of 0 for a major version identifier MUST be limited to initial development.

R7  Version 1.0.0 MUST be used the initial public version.

R8  The major version identifier MUST be incremented when a new public version is created/issued that
    breaks backwards compatibility.

R9  The minor version identifier MUST be incremented when a new public version is created/issued that
    maintains backwards compatibility.

R10 The patch version identifier MUST be incremented when a bug fix is introduced.
Backwards Compatibility
A new version of a Web API is backwards compatible if it does not break clients using a previous version of the API.

R12 Clients **MUST** be designed to ignore data elements that are not recognized.

According to the robustness principle, also known as Postel’s law, which states:

"Be conservative in what you send, be liberal in what you accept."

Client Impacts
- A client, using a previous version of an API, can continue to use the new version of that API, offered on a server, without any negative impact.
A RESTful Web API Design Standard

Section 7.2.3.2, URI Path for API Version

R76 The major version identifier of the Web API **MUST** be represented in the URI path segment.

In the following example, `v1` represents version 1 of the *associates* Web API where the “api” concept is represented in the URI authority.

http://api.abc.com/hr/v1/associates

**Considerations**

- Ideally – no version information in the URI at all.
- Practically – need some way to readily and visible way to identify and request the version of an API.
- Result – put version in the URI, but limit it to the major version identifier.

[Preston-Werner, https://semver.org]
An Operational Model for **RESTful Web API design**

Some considerations in determining an Operational Model for RESTful Web API design...

- Process & Data Enablers
- Organizational Approach
- Technology Enablers

[Richardson et al. (2008)]
## An Operational Model for **RESTful Web API** design

### Process & Data Assets

### Process.

- Gather Requirements
  - Develop integration use cases
- Analyze
  - Develop the Systems Interaction Model
  - Develop the Logical API Model
  - Review
- Design API
  - Develop the API Specification.
  - Verify requirements
  - Verify compliance with Design Standard
- Publish API Specification

### Data Assets.

- Enterprise-level logical domain model.
- Reusable Message Component & Fields Library
- API Specification Templates
- API Specification
An Operational Model for **RESTful Web API design**

**Organizational Approach**

**Centralized API Design Team** comprises integration experts to develop & maintain the enterprise canonical language.

- Receive integration use cases from product stakeholders
- Develop the Systems Interaction Model
- Understand the domain and develop the Logical API Model
- Develop the API Specification and Samples
- Review API Specification with product stakeholders

**Pros**
- Team is intimately familiar with API Design Process, API Design Standards, Tooling, and API Library,

**Cons**
- Team is usually less familiar with a domain
- Doesn't scale well

**Federated API Design Teams** comprises product stakeholders.

- Develop integration use cases
- Develop the Systems Interaction Model
- Develop the Logical API Model
- Develop the API Specification and Samples
- Submit (above) API Specification for review

**Centralized API Governance Team** governs the API Design

**Pros**
- Experts in their domain
- Scales better

**Cons**
- Learning curve for Federated API Teams on API Design Process, API Design Standards, Tooling, and API Library

---

Copyright © 2019 ADP, LLC. Proprietary and Confidential.
An Operational Model for **RESTful Web API design**

Method and Technology Enablers

Some enablers...

- UML Interaction Diagrams for System Interactions Model
- UML Class Diagrams for Logical API Model
- JSON Schema for Messages, Component & Fields Library
- Swagger (OAS) 3.0 for API Specification Template/API Specification
  - Swagger Editor
- GIT Repository/Pull Requests

[Vernoff (2019)]
An Operational Model for **RESTful Web API design**

**Method and Technology Enablers**

Some enablers...

- API Specifications Publication
  - Packaged and Released to the API Registry
  - Available for search and download from the API Hub
  - Used by other systems in the infrastructure.

---

---

https://developers.adp.com

---

[Vernoff (2019)]
Thank You

steffen.fohn@adp.com