{"logo":"API Evangelist"}

API Providers Guide - API Design

Prepared By Kin Lane

June 2014

API DESIGN

Table of Contents

Overview of The API Design Space

In the early days of APIs, everything was just about deploying and consuming—you were doing one or the other. Then by 2006 we saw the stabilization of common API management practices emerge from providers like Mashery, and then 3Scale. Now in 2014 we are stabilizing again, and the API universe is expanding around the area of API design, with new approaches, tools and entire companies emerging to provide services that are dedicated to helping companies design the best APIs they can.

I define the world of API design as everything that goes into planning and designing your API, as well as the definition of your API that will eventually be deployed as your production endpoints, drive your documentation, generate the code samples your developers will use to integrate, test, monitor, and allow your API to be found--through this lens, API design is fast becoming the driver for all areas of the API lifecycle.

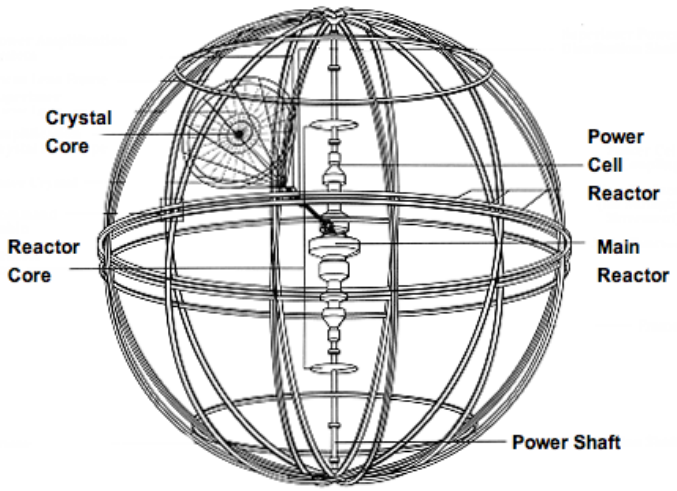
Depending on our needs, API design may begin with understanding HTTP, and REST, or may learning more about applying hypermedia as part of your design constraints, or API design might simply be about generating a Swagger definition, so you can generate interactive API documentation. I won't be going into the finer points of REST, Hypermedia or even Swagger definitions. My goal is to provide a 100K view of space, and highlight the various tools, services, and API design methodologies available today.

Ultimately your API design will be the definition of each endpoint, its methods, fields and much more. In a technical sense it is a JSON, YAML or markdown blueprint that describes your API, and in a creative sense, API design is an art that can possess a strange technical beauty, and speak to the value an API delivers to end-users. API design is far more than just how you structure URIs, or parameters, it is fast becoming a discipline, with a suite of services and tools to support a growing wave of API design professionals.

Thoughtful API design early on can save you a lot of mistakes down the road. This project is not meant to endorse any particular approach or methodology, but provide a single resource where you can find the best information on API design, and go there to find what you need. It can be difficult to stay up with the latest tools, and approaches, and API Evangelist looks to fill this gap.

This API design project will live as an open source repository, and resulting white paper containing the building blocks of API design, tools that assist you in your API design, and companies that provide services in the area of API design.

This is a living project, that will be updated as I have time.. If you know of missing building blocks, or tools and services that should be included, send them to info@apievangelist.com



A New Generation Of API Design

This is not a story of WSDL or even WADL, this narrative beings in early 2011 with the release of a new Wordnik developer area, which included a new version of the API documentation, that wasn't just the same old static API docs--this time it was being driven by a new API definition format, that Tony Tam from Wordnik would later call Swagger. Wordnik needed a way to keep their documentation up to date, so they created Swagger, a JSON format for describing the Wordnik API, and then built Swagger UI, an automatically generated, interactive documentation for the API.

With this one release, Swagger would usher in a new generation of API design, giving us a new way to describe the increasing number of APIs we were developing, and allow us to automatically deliver, interactive documentation that would help consumers understand the value an API delivers. Wordnik didn't stop there, they created tooling for generating server side and client side code, directly from the same Swagger API definitions.

Wordnik had developed Swagger to meet their own needs around designing and maintaing the Wordnik API, but through this process, they also set the stage for a new way to define our APIs, that would give us a a common, machine readable language for discussing and collaborating around our APIs, generating code, documentation, and as time would show, much, much more.

Developing The Language We Need To Communicate

We are still in the infancy of the API economy, and now with barely 14 years of web API design evolution, we are only now just developing the languages we will need to communicate around APIs throughout their lifecycle, from the first mock of the API resource, to monitoring of a production API in the wild, or making available to a new breed of API search engines.

There has long been standards for describing APIs, such as WSDL for SOAP, and WADL for web APIs, but these formats would never actually enable the meaningful interactions around APIs that it would take to achieve widespread support. WSDL was too programmatic, and heavy handed, while WADL never possessed the incentives API providers needed, to take the time to define their APIs using the heavy XML format.

It wasn't until early 2011, we would see the first signs of a meaningful language for describing web APIs, that would enable us to have productive conversations, and take action around our API designs. In February 2011, online word meaning and definition site Wordnik, launched a new API developer portal, complete with a new way to document APIs, that was interactive, and allowed developers to make live API calls, while learning about what an API offered. In an effort to keep their API documentation up to date, Wordnik also established a new way to not just document and communicate with an API, but allow us to actually have a shared conversation about the actual design of the API itself--changing what API design meant.

Later on that year, API management provider Mashery would emulate Wordnik's approach, and launched their I/O Docs. Immediately after, in August 2011, Wordnik formally launched their approach to defining APIs, generate interactive documentation, and server or client side code—which they called Swagger. Over the next two years the Swagger ecosystem would grow and evolve, producing new versions, tooling, and implementations of both private and public API implementations.

During this time, the concept of API design would expand, with tech giants like Google establishing their own approach, which they called Google Discovery, and we would see a new breed of API design service providers emerge with Apiary.io. By March 2013, Apiary had launched their own API definition format called API Blueprint, which allowed API providers to define an API using markdown, and iterate on the design using Apiary.io services. Apiary had moved the incentives for defining APIs as a machine readable format, earlier on in the lifecycle. Now we didn't just generate machine readable definitions to deploy server code, or generate interactive docs and client side code, we would do it before we actually deployed the API, allowing us to mock our API designs, and collaborate around designing the actual API, prior to deployment.

Then by the end of 2013, we'd see another API definition language emerge, this time from enterprise technology provider Mulesoft, called RAML. This new approach to defining APIs, would use YAML to produce a machine readable, yet human friendly way to describe APIs. Now we had four ways to describe our APIs, and as we push into 2014, we also have a wealth of tooling, services, and a selection of maturing API design editors to choose from when planning and designing our APIs.

I feel like we are just moving from infancy, into the toddler phase of communicating throughout the web API lifecycle. We now have several languages to choose from, we just have to work hard to educate people that they exist, assist

them become fluent with their chosen format, and expand the incentives for generating machine readable definitions for our APIs.

Leading API Definition Formats

In the last couple years, we've seen an emergence of four new machine readable formats, that give us a new way to describe APIs, as well as interact with others throughout the APIs lifecycle. The emerging formats use JSON, Markdown, and YAML to define an API, providing a simple, machine readable blueprint, that API providers and consumers can use a single truth for all API interactions.

I'm also looking at other formats I've come across in the space, and will be adding them to my list as I better understand them, and able to quanity their reach. For now I'll be tracking on the four leading API definition formats.

Swagger

The goal of Swagger™ is to define a standard, language-agnostic interface to REST APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined via Swagger, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. Similar to what interfaces have done for lower-level programming, Swager removes the guesswork in calling the service.

API Blueprint

API Blueprint is a documentation-oriented API description language. A couple of semantical assumptions over the plain Markdown. API Blueprint is perfect for designing your Web API and its comprehensive documentation but also for quick prototyping and collaboration. It is easy to learn and even easier to read – after all it is just a form of plain text. API Blueprint, its parser, and most of its tools are completely open sourced so you don't have to worry about vendor lock-in. This also means you can freely integrate API Blueprint into any type of product, commercial or not.

RAML

RESTful API Modeling Language (RAML) is a simple and succinct way of describing practically-RESTful APIs. It encourages reuse, enables discovery and pattern-sharing, and aims for merit-based emergence of best practices. The goal is to help our current API ecosystem by solving immediate problems and then encourage ever-better API patterns. RAML is built on broadly-used standards such as YAML and JSON and is a non-proprietary, vendor-neutral open spec.

Mashery I/O Docs

I/O Docs is a live interactive documentation system for RESTful web APIs. By defining APIs at the resource, method and parameter levels in a JSON schema, I/O Docs will generate a JavaScript client interface. API calls can be executed from this interface, which are then proxied through the I/O Docs server with payload data cleanly formatted (pretty-printed if JSON or XML). Basic HTML text tags are enabled in the JSON schema.

Building Blocks of API Design

When I look at all of the current companies who are developing services and tools for the API design space, I begin to see certain building blocks emerge, providing a kind of common blueprint for quantitying the world of API design. My goal is to help understand what features, tools, and services these companies are delivering, but also which ones API designers are demanding and actually putting to use.

The goal of this research, is to identify common building blocks that can help new, as well as advanced API providers, understand the different approaches to API design, while not making any assumption on which is best, just shed light on what is available, and maybe a look into the pros and cons, and how each of the solutions fit into the bigger world of API design.

Building blocks are about assembling API design concepts into modular, bit-size chunks, in an effort to educate and prepare you, for your own API design efforts. Keeping in line with where I see API design headed, my list of building blocks, naturallys starts with machine readable API definitions.

Definition

A central, machine readable definition of an API interface, authentication and potentially data model, in XML, JSON or Markdown. (Examples: API Blueprint, RAML, Swagger)

Parser

An API definition parser, available potentially in multiple languages and open up the programmatic generation of other API building blocks.

Design Tools

User interface tools, allowing for the building of central API definitions, either in a code view or GUI view.

Versioning

Systems allowing for the versioning of API definition, keeping track of all changes, allowing for rolling back of changes to previous versions.

Forkable

The ability to fork an existing API definition, and create a new branch, that can live separately from the API definition it originates from.

Sharing

Allowing for the sharing of API definitions and other API design building blocks with other users, employing common social sharing features of preferred networks.

Collaboration

Features that allow for collaboration between users, with discussion around all API design building blocks.

Mock Interfaces

Ability to deploy mock API interfaces generated from API definitions, allowing developers to play with API versions as they are designed.

Interactive Documentation / Console

Automatically generated API documentation which allows developers to make calls against APIs as they are learning about the interface, turning API education into a hands on experience.

Notebook / Directory

A local, or cloud based storage repository, providing a single place to create and manage API definitions, and execute other API design building blocks.

Testing

Manual, automated and scheduled testing of API interfaces using their API definition as a blueprint, allowing developers to test all APIs to make sure they comply to API definition.

Debugging

Manual, automated and scheduled debugging of API interfaces, providing detailed look inside of API calls, allowing developers to understand problems with API integrations.

Traffic Inspection

Logging and analysis of API traffic from testing, debugging and all other API usage during the API design process.

Validator

Tools for validating API calls, enabling developers to determine which types of calls will be valid, using central API definition.

Server Code Generators

Tooling that generates server side implementations using API definitions in a variety of languages.

Client Side Code Generator

Tooling that generates client side API code libraries in a variety of languages.

Github Sync

The ability to store and sync API definitions with Github, providing a central public or private repository for the definition of an API resource.

Command Line

Command line tooling for execution of all API building blocks.

Websockets

Providing tools for API communication via websockets using API definition as a guide.

Translator

Tools for translating between various API definitions, allowing the transformation from RAML to Swagger, and between each API definition format.

Annotation

Tools and interfaces for allowing the annotation of API definitions, providing a communication platform centered around the API design process.

Syntax Highlight

Tools and interfaces for the highlighting of API definitions, providing IDE-like functionally for API designers.

Companies Who Provide API Design Services

Most API design work is done by companies in support of their own API programs, but as the space expands there are new companies emerging, providing us with new tools, services, and approahces specifically dedicated to API design. I'm working to keep track of these companies, and understand how they are influencing the space--here are the four I'm currently tracking on.

apiary.io

https://apiary.io
http://blog.apiary.io/
http://blog.apiary.io/feed.xml
https://twitter.com/apiaryio
https://github.com/apiaryio
http://crunchbase.com/company/apiary

Apiary.io

Apiary.io is a hosted suite of tools that help companies build web APIs quickly, test & monitor them easily and document them effortlessly. It provides API owners with necessary infrastructure and helps them build relationship with their users. Core of the self-service solution is API Blueprint, an efficient format for describing an API, aspiring to define new gold standard for REST API development. The product uses this Blueprint to streamline adoption of new services and simplify integration to other systems.

http://mashery.com
http://www.mashery.com/blog
http://feeds.feedburner.com/MasheryBlog
https://twitter.com/Mashery
https://github.com/mashery
http://crunchbase.com/company/mashery

Mashery

Mashery's API management tools and strategic services help companies connect with customers and partners in a changing digital world by extending reach across devices, markets and the Web. Mashery leads the industry with a holistic approach for API initiatives – from setting platform strategy and measuring business objectives to the heavy lifting of providing and managing infrastructure to facilitating relationships with our 150,000 strong network of Web and mobile application developers. Having worked with over 100 leading brands to power more than 40,000 apps, Mashery's knowledge, experience and proven strategies enable companies to focus on their core business while driving sales, building new revenue channels and realizing faster time-to-market for innovative applications.

http://www.mulesoft.com/
http://blogs.mulesoft.org/
http://feeds.feedburner.com/muleblog
https://twitter.com/MuleSoft

MuleSoft

MuleSoft provides the most widely used integration platform for connecting SaaS and enterprise applications in the cloud and on-premise. With the rise of cloud and mobile, enterprises face a choice: get overwhelmed by the resulting explosion of end points or seize the opportunity to gain competitive advantage. Founded on the idea that connecting applications should not be hard, MuleSoft lets organizations harness the power of their applications through integration. MuleSoft's Anypoint™ technology eliminates costly, time-intensive point-to-point integration, enabling business agility. Delivered as a packaged integration experience, CloudHub™ and Mule ESB™ (Enterprise Service Bus) are built on proven open source technology for the fastest, most reliable on-premise and cloud integration without vendor lock-in.

Swagger

Swagger UI is part of Swagger project. The Swagger project allows you to produce, visualize and consume your OWN RESTful services. No proxy or 3rd party services required. Do it your own way. Swagger UI is a dependency-free collection of HTML, Javascript, and CSS assets that dynamically generate beautiful documentation and sandbox from a Swagger-compliant API. Because Swagger UI has no dependencies, you can host it in any server environment, or on your local machine.

http://swagger.wordnik.com/

https://github.com/Swagger

API Design Tools

There are many new tools emerging from the API design work going on across the API sector in 2014. Right now, many of these API tools seem to be focused on generating API definitions in various formats, including JSON, YAML and Markdown, with a focus on API deployment, manageent, and more recently integration and discovery.

Generally these tools are open source repositories, that are stored at Github. My goal is to empower you, the API provider or consumer, with the resource you will nee to grow and evolve your API design strategy.

I tend to organize any interesting tools I find under a single umbrella, but as I start to see common patterns, I will break them out into separate groups--such as with the API design editors listed below.

ALPS - Application-Level Profile Semantics

http://amundsen.com/hypermedia/profiles/

The purpose of Application-Level Profile Semantics (ALPS) is to document the application-level semantics of a particular implementation. This is accomplished by describing elements of response representations for a target media type. For example identifying markup elements returned (i.e. semantic HTML ala Microformats) and state transitions (i.e. HTML.A and HTML.FORM elements) that advance the state of the current application.

Atom Editor API Blueprint Preview

https://atom.io/packages/api-blueprint-preview

A plugin for the Atom editor to show the rendered HTML API Blueprint to the right of the current editor using ctrl-shift-a.

RAML API Designer

https://github.com/mulesoft/api-designer

API Designer is a web-based API development tool that allows API providers to design their API quickly, efficiently, and consistently, and socialize the design. It consists of a RAML editor side-by-side with an embedded RAML console (the API Console). It is provided under the open-source CPAL license.

RAML API Notebook

https://github.com/mulesoft/api-notebook

API Notebook is a web-based, persistent, JavaScript scripting workspace that

enables live testing and exploring of APIs, and saving API use cases as markdown gists, so they are versioned, forkable and shareable. It's an example of literate programming. It is provided under the open-source CPAL license.

RAML Store

https://github.com/brianmc/raml-store

RAML Store provides a simple storage API plus a persistence plugin which enables you to run the RAML API Designer locally (rather than using a cloud service) and still be able to manage and collaborate on your design. The service is built with node.js, using express and mongodb.

Rspec APIBlueprint

https://github.com/playround/rspec_api_blueprint

Autogeneration of API documentation using the Blueprint format from request specs.

Swagger Editor

http://editor.swagger.wordnik.com/

Swagger Editor lets you edit API specifications in YAML inside your browser and to preview documentations in real time. Valid Swagger JSON descriptions can then be generated and used with the full Swagger tooling (code generation, documentation, etc).

Swagger2RAML

https://github.com/8x8Cloud/swagger2raml

A utility to generate RAML documentation from Swagger JSON.

API Design Editors

Listed above in tools you will find two open source API design editors, from RAML and Swagger. I'd like to break out API design editors into its own grouping, because I feel this a significant area of growth in the API design space over the last year, and will continue to be a focal point for the evolution of API design.

The first API design editor was as a service, from Apiary.io, but last year we also saw the rollout of the RAML API Designer, and this last month Swagger rolled out their own editor, designed around the Swagger specification.

Apiary.io

https://apiary.io

Apiary.io is a hosted suite of tools that help companies build web APIs quickly, test & monitor them easily and document them effortlessly. It provides API owners with necessary infrastructure and helps them build relationship with their users. Core of the self-service solution is API Blueprint, an efficient format for describing an API, aspiring to define new gold standard for REST API development. The product uses this Blueprint to streamline adoption of new services and simplify integration to other systems.

RAML API Designer

https://github.com/mulesoft/api-designer

API Designer is a web-based API development tool that allows API providers to design their API quickly, efficiently, and consistently, and socialize the design. It consists of a RAML editor side-by-side with an embedded RAML console (the API Console). It is provided under the open-source CPAL license.

Swagger Editor

http://editor.swagger.wordnik.com/

Swagger Editor lets you edit API specifications in YAML inside your browser and to preview documentations in real time. Valid Swagger JSON descriptions can then be generated and used with the full Swagger tooling (code generation, documentation, etc).

API Definitions Providing A Central Truth For The API Lifecycle

As these new ways of communicating around APIs emerge, and evolve, they are providing us with a single definition of an API, that can be applied as a sort of truth, through all stages of the API's lifecycle.

While machine readable API definitions may not deliver on automating all aspects of the API lifecycle (as some envision), it does provide a set of guiding principles, acting as a truth when mocking, deploying, generating code, testing, monitoring, producing documentation and other API content.

The recent API design era we are in was kicked off by the desire to generate interactive documentation, but in 2014 we are seeing more reasons for generating machine readable API definitions, and using them as a guide throughout the API lifecycle.

Beyond a single definition that can be used throughout a single APIs lifecycle, by the API provider, API definitions also allow for opening up a conversation externally with API consumers, and other 3rd party developers, using a common language and syntax--establishing a potentially shared meaning around what an API delivers, internally and externally.

The single most important thing that machine readable API definitiosn provide, is the ability to share successful, or common API blueprints with the wider API sector, or just within a specific industry. A catalog of well designed API blueprints, that API providers can use, has long been a deficiency across the API space--API definitions provide a truth, allowing for the establishment, sharing and re-use of the best API patterns available, allowing us to not re-invent the wheel with every API--increasing interoperability and potential for automation.

Contributing To The Deployment Lifecycle

Automatically generating an API, from a machine readable definition, has long been the dream of API providers. In reality, this is much harder to achieve, than one might think, but there has been significant work from leading API design service providers and their developer communities, and in 2014, you can indeed auto generate an API from your API definition.

API definitions like API Blueprint, RAML, or Swagger, do what they do best--describe your interface. How this merges with the actual deployment of an API backend, is not always evident. Depending on how you craft your API definition, it may contain your underlying data model, and for simpler data or content APIs, auto generation of a backend might be possible.

When it comes to more complex API resources, there will undoubtedly be much more magic behind the interface, requiring connection with unique backend systems, and code libraries. I'm confident that the API community will continue to produce API definition driven, connectors for common IT systems, further evolving on the vision of automatically generating of APIs from their machine readable API definitions—replacing much of what more traditional service gateways have always delivered.

There is also a new evolution in cloud computing, one that will contribute significantly to API deployment, being called containers. New approaches to deploying application resources, from Docker, and subsequently Amazon, Red Hat, Google, Microsoft, and others, will rapidly escalate how we deploy the APIs that we depend on. API definitions provide a perfect interface definition, for resources that are deployed as containers. We are just missing the linking between current container deployment definitions, and the evolving world of API definitions—a link that will come into focus in 2014.

Contributing To The Management Lifecycle

The early motivation for generation machine readable definition using Swagger, was the promise of being able to keep your API documentation up to date, while also providing users with an interactive way to explore and learn about your API. Quickly API providers also developed tooling for generating of client side API libraries in all the popular programming languages, including but not limited to PHP, Python, Ruby, JavaScript, Node.js, Java, C# and Go.

Machine readable API definitions can also be used to generate permanent or temporary sandbox environments for API consumers, allowing for environments that developers can learn about an API, before they put the API to use in a real production environment. Like documentation, sandbox environments can be costly and time consuming to deploy and keep up to date, and API definitions can help make it much more painless and automated.

As the world of API design expands, I think we should stop regularly, and take a look at our API management workflow, to see how a central API definition can help manage or even automate other areas of our API operations. While we will never automate all aspects of API management, API definitions can provide a central truth that can be applied in many unexpected ways.

Contributing To The Testing & Monitoring Lifecycle

When it comes to testing, and monitoring an API, you begin to really see how machine readable API definitions can be the truth, in the contract between API provider and consumer. API definitions are being used by API testing and monitoring services like SmartBear, providing a central set of rules that can ensure your APIs deliver as promised.

Making sure all your APIs operate as expected, and just like generating up to date documentation, you can ensure the entire surface area of your API is tested, and operating as intended. Test driven development (TDD) is becoming common practice for API development, and API definitions will play an increasing role in this side of API operations.

An API definition provides a central truth, that can be used by API providers to monitor API operations, but also give the same set of rules to external API monitoring services, as well as individual API consumers. Monitoring, and understanding an API up time, from multiple external sources is becoming a part of how the API economy is stabilizing itself, and API definitions provide a portable template, that can be used across all API monitoring services.

Testing and monitoring of vital resources that applications depend on is becoming the norm, with new service providers emerging to assist in this area, and large technology companies like Google, making testing and monitoring default in all platform operations. Without a set of instructions that describe the API surface area, it will be cumbersome, and costly, to generate the automated testing and monitoring jobs necessary to produce a stable, API economy.

Contributing To The Discovery Lifecycle

One of the newest incentives for API providers to develop machine readable API definitions, is about API discovery. After many years of no search solution for APIs, beyond the standard API directory, we are just now beginning to see a new generation of API discovery tooling and services.

Together with 3Scale, I recently launched the APIs.json discovery format, looking to create a single API discovery framework that can live within the root of any domain. Our goal is to allow API providers to describe their APIs, providing machine readable pointers to the common building blocks of a company's API like signup, machine readable definitions, documentation, terms of service, and much, much more.

As we were developing APIs.json, we recognized that without a proper, distributed search engine, any machine readable API discovery formats would not be successful, and with this in mind 3Scale launched an open source API search engine, with the first implementation being APIs.io. As the number of APIs rapidly grows, more search solutions like APIs.io will be needed to make sure the most valuable APIs are discoverable, in real-time.

The future of API discovery will need more than just basic metadata to find APIs, we will need machine readable definitions that describe the API, as well as its supporting building blocks. API definitions will help automate the discovery and understanding of what value an API delivers, helping API consumers find just the API resources they need to make their applications successful.

An Evolutionary Period For API Design

Good API design practices, tools and services have a major impact downstream within API initiatives. As more companies reach a maturity point with their API efforts, the desire to refine API design before incurring the costs associated with deployment and management, while working to mitigate risks, will only increase.

API design has been historically hitched to RESTful dogma and early SOA practictioners vision of API discovery. As the web API world matures, API design is also becoming about deployment, documentation, code samples, testing, monitoring, and discovery, and other essential building blocks of a healthy API lifecycle.

API design should be first and foremost about delivering quality, consistent and meaningful API endpoints, while also delivering value for management, discovery and integration. In coming years, we will see specific industries realize their role in leading API design--energy, healthcare, transportation, environment, and other industries will help define meaningful API design standards for their industries through partnerships, government leadership, and most importantly, successful patterns established by existing API leaders.

My research project into API design is meant to identify the evolving building blocks, tools and companies that are contributing to the success of the API design sector. More resources are provided below as an appendix, but you can also find everything published to Github at **design.apievangelist.com,** while also being published as this white paper to provide the largest possible reach for this valuable information.

The API industry has come of age, and things are rapidly evolving when it comes to defining best practices and providing tools that will assist companies and developers achieve the best possible API design--putting us in a very important period for the API economy.

{"logo":"API Evangelist"}

www.apievangelist.com

info@apievangelist.com

Appendix A: Curated News and Resources

- **API First Development with RAML and SoapUI [Webinar]** from feedproxy.google.com on 6/19/2014), full resource available at http://feedproxy.google.com/~r/muleblog/~3/SYHjihCKVqI/
- **Introducing The RESTed NARWHL - A Practical API Design Framework** from feedproxy.google.com on 6/17/2014), full resource available at http://feedproxy.google.com/~r/MasheryBlog/~3/b3GnPh-QIvk/introducing-rested-narwhl-practical-api-design-framework
- **Happiness is a well-designed API | ZDNet** from www.zdnet.com on 6/6/2014), full resource available at http://www.zdnet.com/happiness-is-a-well-designed-api-7000030282/
- **Swagger Levels The API Design Playing Field With New Editor And YAML Definitions** from apievangelist.com on 6/5/2014), full resource available at http://apievangelist.com/2014/06/05/swagger-levels-the-api-design-playing-field-with-new-editor-and-yaml-definitions/
- **API documentation made beautiful with Apiary.io | ITworld** from www.itworld.com on 2/28/2014), full resource available at http://www.itworld.com/development/407333/api-documentation-made-beautiful-apiaryio
- **A Practical… by D. Keith Casey Jr et al. [PDF/iPad/Kindle]** from leanpub.com on 2/27/2014), full resource available at https://leanpub.com/restful-api-design
- **Application Programming eXperience: It's all about *X - Mobile Apps Stuff** from manfredbo.tumblr.com on 1/28/2014), full resource available at http://manfredbo.tumblr.com/post/74280033341/application-programming-experience-its-all-about-x
- **Jakub Nesetril, CEO of Apiary on Web APIs and Developer Experience** from www.infoq.com on 1/28/2014), full resource available at http://www.infoq.com/interviews/jakub-apis-dx
- **The Human Aspects of API Design: An Interview with Apiary's Jakub Nesetril** from www.infoq.com on 11/14/2013), full resource available at http://www.infoq.com/news/2013/11/apiary-jakub-nesetril-interview
- **More Thoughts on an API Commons** from www.3scale.net on 11/10/2013), full resource available at http://www.3scale.net/2013/11/towards-api-commons-for-the-api-economy/
- **Designing APIs for the Internet of Things (IoT)** from www.layer7tech.com on 10/30/2013), full resource available at http://www.layer7tech.com/blogs/index.php/designing-apis-for-the-internet-of-things-iot/
- **APIs At The Heart of your Mobile App Strategy** from blog.soa.com on 10/28/2013), full resource available at http://blog.soa.com/apis-at-the-heart-of-your-mobile-app-strategy/
- **No more outdated API documentation!** from blog.apiary.io on 10/10/2013), full resource available at http://blog.apiary.io/2013/10/10/No-more-outdated-API-documentation/
- **RAML - RESTful API modeling language** from raml.org on 10/2/2013), full resource available at http://raml.org/index.html
- **New API Blueprint available at Apiary** from blog.apiary.io on 10/2/2013), full resource available at http://blog.apiary.io/2013/10/02/New-API-Blueprint-available/
- **Apiary Is Growing** from blog.apiary.io on 9/17/2013), full resource available at http://blog.apiary.io/2013/09/17/Apiary-Is-Growing/
- **JSON and XML** from developer.infoconnect.com on 9/3/2013), full resource available at http://developer.infoconnect.com/json-and-xml
- **MASHERY I/O DOCS - DOCS THAT ROCK** from www.mashery.com on 8/27/2013), full resource available at http://www.mashery.com/blog/mashery-io-docs-docs-rock
- **Designing APIs for Asynchrony** from blog.izs.me on 8/24/2013), full resource available at http://blog.izs.me/post/59142742143/designing-apis-for-asynchrony
- **Pluralsight - Web API Design with Shawn Wildermuth - E-Books + Tutorials...** from skdown.net on 8/21/2013), full resource available at http://skdown.net/forum/index.php/showtopic=4317200
- **Api Design, Part 1: Rest Is The Leading But Not Only...** from www.forrester.com on 8/20/2013), full resource available at http://www.forrester.com/API+Design+Part+1+REST+Is+The+Leading+But+Not+Only+Option+For+Your+APIs/fulltext/-/E-RES102021
- **Api Design, Part 2: Design Messaging Styles By Balancing...** from www.forrester.com on 8/20/2013), full resource available at http://www.forrester.com/API+Design+Part+2+Design+Messaging+Styles+By+Balancing+Reach+With+Your+Other+Desig
/E-RES102022
- **Api Design, Part 3: Make Transactions And Error Handling...** from www.forrester.com on

8/20/2013), full resource available at http://www.forrester.com/API+Design+Part+3+Make+Transactions+And+Error+Handling+Clear+In+Your+API+Designs/full
/E-RES102023

- **Api Design, Part 4: Future-proof And Secure Your Apis...** from www.forrester.com on 8/20/2013), full resource available at http://www.forrester.com/API+Design+Part+4+FutureProof+And+Secure+Your+APIs+To+Fit+Your+Usage+Scenarios/fullte
/E-RES102024
- **The Secrets of Awesome JavaScript API Design** from css.dzone.com on 8/16/2013), full resource available at http://css.dzone.com/articles/secrets-awesome-javascript-api
- **The Importance of Impermanence in API Design** from www.programmableweb.com on 8/6/2013), full resource available at http://www.programmableweb.com/2013/08/06/the-importance-of-impermanence-in-api-design
- **Joshua Bloch: Bumper-Sticker API Design** from www.infoq.com on 7/27/2013), full resource available at http://www.infoq.com/articles/API-Design-Joshua-Bloch
- **Micro Service Architecture** from yobriefca.se on 7/9/2013), full resource available at http://yobriefca.se/blog/2013/04/29/micro-service-architecture/
- **ADL's Experience API Design Working Group Kick-Off on July 15** from elearningindustry.com on 7/9/2013), full resource available at http://elearningindustry.com/adl-s-experience-api-design-working-group-kick-off-on-july-15
- **API Design: A New Model for Pragmatic REST** from blog.apigee.com on 7/4/2013), full resource available at https://blog.apigee.com/detail/api_design_a_new_model_for_pragmatic_rest
- **API Design: Harnessing HATEOAS, Part 2** from blog.apigee.com on 7/3/2013), full resource available at https://blog.apigee.com/detail/api_design_harnessing_hateoas_part_2
- **Signs you're veering from good API design** from blog.apiaxle.com on 7/3/2013), full resource available at http://blog.apiaxle.com/post/signs-youre-veering-from-good-api-design/
- **API Crafting Secrets: Into Flightstats APIs** from www.3scale.net on 7/2/2013), full resource available at http://www.3scale.net/2013/07/api-crafting-secrets-into-flightstats-apis/
- **Practical API Design: Confessions of a Java Framework Architect** from www.tinydl.com on 6/29/2013), full resource available at http://www.tinydl.com/1060908845-practical-api-design-confessions-of-a-java-framework-architect.html
- **Better Rest API design 1** from hao-deng.blogspot.com on 6/27/2013), full resource available at http://hao-deng.blogspot.com/2013/06/better-rest-api-design-1.html
- **API Design: Harnessing HATEOAS, Part 1** from blog.apigee.com on 6/20/2013), full resource available at https://blog.apigee.com/detail/api_design_harnessing_hateoas_part_1
- **When Good API Design is a Waste of Time** from www.layer7tech.com on 6/19/2013), full resource available at http://www.layer7tech.com/blogs/index.php/when-good-api-design-is-a-waste-of-time/
- **API Design: Honing in on HATEOAS** from blog.apigee.com on 6/17/2013), full resource available at https://blog.apigee.com/detail/api_design_honing_in_on_hateoas
- **RESTful APIs: White House Sets The Standard(s)** from www.programmableweb.com on 6/17/2013), full resource available at http://www.programmableweb.com/2013/06/17/restful-apis-white-house-sets-the-standards/
- **A Hypermedia API Reading List - Literate Programming** from blog.steveklabnik.com on 6/12/2013), full resource available at http://blog.steveklabnik.com/posts/2012-02-27-hypermedia-api-reading-list
- **Atlassian REST API Design Guidelines version 1 - Documentation** from developer.atlassian.com on 6/12/2013), full resource available at https://developer.atlassian.com/display/DOCS/Atlassian+REST+API+Design+Guidelines+version+1
- **Designing Hypermedia APIs** from www.designinghypermediaapis.com on 6/11/2013), full resource available at http://www.designinghypermediaapis.com/
- **API Design Wiki** from wiki.apidesign.org on 6/11/2013), full resource available at http://wiki.apidesign.org/wiki/Main_Page
- **Web API Design Cookbook** from www.w3.org on 6/11/2013), full resource available at http://www.w3.org/TR/api-design/
- **Swagger with WSO2 API Manager** from blog.cobia.net on 5/31/2013), full resource available at http://blog.cobia.net/cobiacomm/2013/05/31/swagger-with-wso2-api-manager/
- **API Hierarchy of Needs | API UX** from apiux.com on 5/30/2013), full resource available at http://apiux.com/2013/05/29/api-hierarchy-needs/
- **Stop Designing Fragile Web APIs by Mathieu Fenniak** from mathieu.fenniak.net on 4/29/2013), full

resource available at https://mathieu.fenniak.net/stop-designing-fragile-web-apis/

- **White House API Standards** from github.com on 4/26/2013), full resource available at https://github.com/WhiteHouse/api-standards
- **Three Ways to Think About API Design** from www.programmableweb.com on 4/26/2013), full resource available at http://www.programmableweb.com/2012/02/14/three-ways-to-think-about-api-design/
- **REST API Design Rulebook** from shop.oreilly.com on 4/26/2013), full resource available at http://shop.oreilly.com/product/0636920021575.do
- **API Design and Architecture Boot Camp from Layer 7** from www.layer7tech.com on 4/26/2013), full resource available at http://www.layer7tech.com/services/api-design-and-architecture-boot-camp
- **API Design and Documentation** from www.howto.gov on 4/26/2013), full resource available at http://www.howto.gov/digitalgovsplash.php/redirect_to=category/code/api/
- **API Design from Apigee** from apigee.com on 4/26/2013), full resource available at http://apigee.com/about/api-best-practices/api-design/all
- **How to Design a Good API & Why it Matters** from www.infoq.com on 4/26/2013), full resource available at http://www.infoq.com/presentations/effective-api-design
- **Designing APIs for Humans** from john-sheehan.com on 4/25/2013), full resource available at http://john-sheehan.com/blog/designing-apis-for-humans
- **APIs as Art** from apievangelist.com on 3/19/2013), full resource available at http://apievangelist.com/2013/03/19/apis-as-art/
- **API Design & Development Guidelines** from www.dzone.com on 2/16/2013), full resource available at http://www.dzone.com/links/r/api_design_amp_development_guidelines_selected_re.html
- **API Design from Apigee - 3rd Edition** from blog.apigee.com on 1/16/2013), full resource available at https://blog.apigee.com/detail/api_design_third_edition_video_slides
- **OData and Impact on API Design (video & slides)** from blog.apigee.com on 6/3/2012), full resource available at https://blog.apigee.com/detail/odata_and_impact_on_api_design_video_slides/
- **API Design by Matt Gemmell** from mattgemmell.com on 5/24/2012), full resource available at http://mattgemmell.com/api-design/