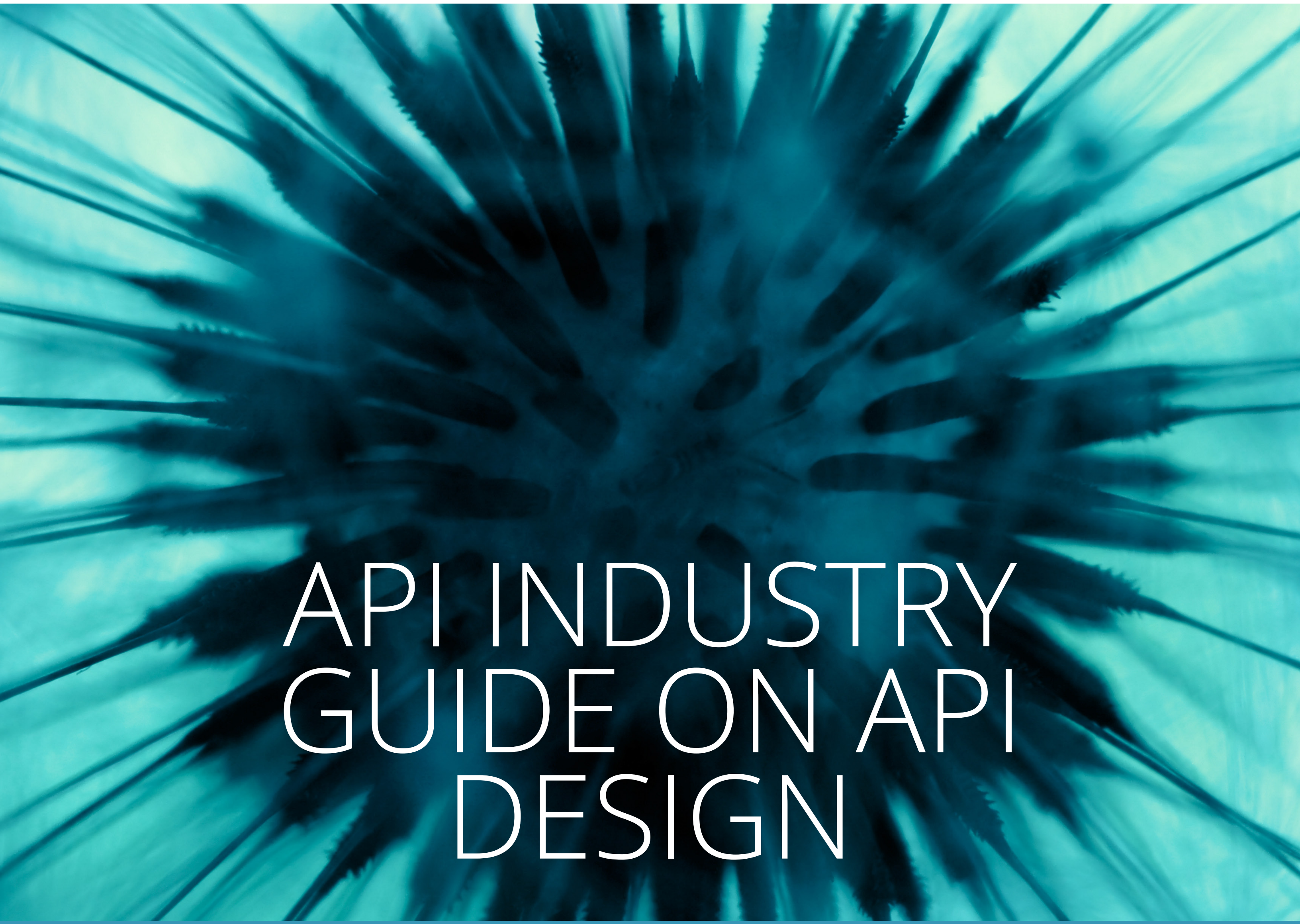
3scale by Red Hat® | {"logo":"API Evangelist"}



API INDUSTRY GUIDE ON API DESIGN

JULY 2015 ■ BY KIN LANE, THE API EVANGELIST

This is meant to be a field guide to the fast-changing world of API design, providing you an overview of companies, tooling, common building blocks, and some of the latest news from across the landscape.

INTRODUCTION TO THE WORLD OF API DESIGN

The concept of API design has evolved rapidly in just the last year. In 2010, the common focus across the sector was API management. However, in 2015, the focus has spread to many stops along the API lifecycle, including much more thoughtful approaches to the design of both internal and external API resources. This guide looks to take a snapshot of this fast-moving space by breaking down the key companies who are playing a role in pushing forward the API design conversations, as well as some of the common building blocks and API design tooling that has emerged.

As the modern web API movement evolved between 2005 and 2012, the concept of API design was left to the API developers who were in charge of crafting the actual server-side code for each API. During this time, there were many API design conversations occurring, but they were limited to API developer circles. Then in 2011, a new company came along that began to shift the conversation of API development and management to earlier on in the lifecycle. This new API design focused company was Apiary.

Apiary provided a web-based, IDE-like environment where API providers could design APIs, then deploy mock interfaces for use in building prototypes without ever actually launching a real API. This opened up a whole new approach to crafting APIs, allowing API architects to share potential API endpoints with developers, and encouraging them to play with them before any design was actually finalized. Apiary also released an API definition format called API Blueprint, which allowed API developers to define the surface area of any API using simple Markdown, establishing a much more human way to design, share, communicate, and collaborate throughout the API design process.

The Apiary team changed how we give birth to API resources. In 2012, another API definition format called Swagger had also emerged, giving us another format for defining APIs. However, the Swagger approach was first focused on generating interactive API documentation and generating client or server code. Apiary focused on enabling the conversation around API design before we needed to generate server code and documentation. Apiary also provided API designers with interactive documentation and the generation of client-side code samples, but the platform's focus was on the design, collaboration, communication, and mocking of API resources – providing a very rich, API-design-first way of operating.

As I do with all my research, I want to better understand what organizations like Apiary are doing, what services they offer, what tooling they've developed, and more about some of the common building blocks they're using to push the API design discipline forward. Hopefully, when done with this paper, we'll both have a better understanding of just what API design is, and take away some new ideas, services, and tools we can all use in our own API operations.

ORGANIZATIONS MAKING AN IMPACT ON API DESIGN

As I conduct my research, I often find it difficult to draw clear lines between the different areas that make up the API space, but I do my best to keep a clear focus for each of my areas. When it comes to API design, I originally had just Apiary on this list, then I quickly added Mashape, and then Mulesoft to the conversation as they introduced their design components. As we moved through 2014 Restlet and Swagger stepped up with more API design-focused services and tooling as well, and then I also stumbled across the approach of Sandbox and Deployd.

While Apigee had been much of the force behind the Swagger API design tooling, at Gluecon this last May they formally released the Apigee API Studio. As API design continues its expansion, the APIMATIC team also saw the importance of having their own API design environment, something I think all API service providers will realize is necessary, providing developers with a doorway to the API lifecycle.

This may not be a complete list of organizations who are leading the API design movement, but as of July 2015 it does represent the core pack. Each of the providers listed below have their own motivations for providing API design tools and services, making for a very rich, and potentially healthy playing field when it comes to API design.

- **Apiary** - Apiary jump-started the modern API design movement by making API definitions more than just about API documentation, allowing API designers to define APIs in the machine-readable API definition format API blueprint, then mock, share, and publish documentation via a cloud platform.

Website:	apiary.io
Twitter:	@apiaryio
Blog:	blog.apiary.io
GitHub:	apiaryio

- **Apigee API Studio** - Apigee launched their API Studio out of their earlier Apigee-127 product, their work on the Swagger platform and editor, and their BaaS offering, opening up the ability for developers to design, mock, test, and share via the online platform. While Apigee Studio is part of the larger Apigee line of products, it is a separate standalone, open source based studio.

Website:	apistudio.io

- **APIMATIC** - When you use APIMATIC to manage SDKs, they provide you with an editor for adding, editing, and deleting the details of each API. When you bundle this with their multi-format API definition format import and export, the platform quickly becomes an API design tool as well as a platform for generating your SDKs.

Website:	www.apimatic.io
Twitter:	@apimatic

- **Deployd** - Deployd is an open source API design and deployment platform that allows developers to quickly design, customize, and deploy an API, with supporting application interface via a downloadable app and command line utilities. Deployd is a downloadable solution you can use on your desktop.

Website:	deployd.com
Twitter:	@deploydapp
GitHub:	deployd

- **Mashape** - Mashape provides an API editor as part of their API management and discovery platform, allowing API providers to add, edit, and manage the details of an API design, while also managing the rest of API operations – from design to discovery and integration. Mashape editor is just a piece of the overall Mashape suite of API lifecycle management tooling.

Website:	mashape.com
Twitter:	@mashape
Blog:	blog.mashape.com
GitHub:	mashape

- **MuleSoft** - Mulesoft provides a cloud and open source version of their API design editor, enabling API designers to craft APIs using the RAML API definition format, then publish to notebook, as well as being manageable through other aspects of the API lifecycle with other Mulesoft systems.

Website:	www.mulesoft.com
Twitter:	@mulesoft
Blog:	blogs.mulesoft.org
GitHub:	mulesoft

- **Restlet** - The Restlet Studio allows you to design APIs and add, edit, and manage the details via a cloud-based API editor, import and export in Swagger and RAML, then also generate server and client-side code, as well as interactive documentation. Restlet Studio is a separate open source project, but is bundled alongside the Restlet framework and API Spark cloud platform.

Website:	apispark.com
Twitter:	@apispark
Blog:	blog.restlet.com
GitHub:	apispark

- **Sandbox** - Sandbox provides an environment for users to quickly mock APIs that are generated from common API definition formats like API Blueprint and Swagger, then deploy, collaboratively build, and debug APIs using an online platform. Sandbox reflects a new type of API definition-driven service providers who use Swagger and API Blueprint to onboard APIs.

Website:	getsandbox.com
Twitter:	@_getsandbox
Blog:	blog.getsandbox.com

- **Swagger** - Swagger is a machine-readable API definition format that has built a number of tools around the specification, including an open source API design editor that allows you to design, import, and export APIs in JSON and YAML, then also generate server and client-side code, as well as interactive documentation.

Website:	swagger.io
Twitter:	@swaggerapi
Blog:	swagger.io/blog
GitHub:	swagger-api

Some of these companies I've listed provide online, cloud-based services, which are reflected below in the common building blocks I outlined, while others also provide open source tooling and other resources included in the API design tooling section. While I wish to share companies and their services with you, a big part of my mission is to better understand the impact each company makes on the industry, resulting in me distilling these companies down to common building blocks, as well as identifying the valuable open source tooling.

One area of my research that I struggle with is around just what exactly is a company, or organization. Some entities I track are businesses, some are government agencies, some are non-profit organizations, while others emerge out of open source tooling that have almost taken on a life of their own. An example of this is Swagger – while Swagger was the brainchild of Wordnik, which is now owned now by SmartBear, the API definition format and open source tooling has established its own elevated presence.

Ultimately I'd say the API design space and my research is still very young, and volatile. While the API design space doesn't have as many players as other areas of deployment and management, I'd like to note that it is one of the fastest growing spaces in the API world currently. I expect this space to double and triple in size over the next year.

SOME OF THE COMMON BUILDING BLOCKS THAT HAVE EMERGED

As I make way through the websites, and GitHub accounts of the companies and organizations listed above, I work to try and pull out some of the common building blocks that make up the API design sector. Like the rest of this research, these building blocks may not be precise, but they do help us understand some of the successful approaches being applied. My goal is to understand, not dictate what is, and building blocks are my legos in this exercise.

API design begins with an idea, but you quickly need a way to articulate exactly what is your API throughout the design process, and throughout every other step in the API lifecycle. While it's easy to think about API design as something you only do previous to the birth of a new API, in reality it's a consideration throughout the API lifecycle, feeding a roadmap that API design editors are providing a doorway to, while also providing essential services and tooling for empowering the API architect at every stage.

Here are the building blocks I've extracted as part of my API design research, providing a base set of components to consider when thinking about what is possible with API design.

- **Definition** - A central, machine-readable definition of an API interface, authentication, and potentially data model in XML, JSON, or Markdown (e.g., WADL, API Blueprint, RAML, Swagger). These formats provide a way to define and communicate around the surface area of an API.

- **Parser** - An API definition parser, available potentially in multiple languages, opens up the programmatic generation of API definitions for use throughout the API lifecycle, starting with API definitions-driven design.

- **Editor** - User interface tooling, allowing for the building of central API definitions, either in a code view or GUI view. API design editors allow users to directly edit definitions, and opens up access to other features, services, and tooling via the IDE like environment.

- **Versioning** - Systems allowing for the versioning of API definition, keeping track of all changes, allowing for rolling back changes to previous versions. It's common for versioning features to sync with all other areas of API lifecycle, through a common approach to versioning in the API definition.

- **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. It's common for API designs to be evolved very similar to code, making GitHub a very attractive base for any API design services and tooling.

- **Sharing** - Encouraging the sharing of API definitions and other API design building blocks with other users, employing common social sharing features of preferred networks. Sharing introduces the essential human element into the API design process, saving potentially costly resources downstream by addressing issues during the design portion of lifecycle.

- **Collaboration** - Features that allow for and enable collaboration between users, with discussion around the API design process, encouraging designers to seek feedback from business stakeholders while also getting the sign-off from developers before code is even written.

- **Mocking** - Opening up the ability to deploy mock API interfaces generated from API definitions, allowing developers to play with API versions as they're designed. This building block delivers on the promise that API design first allows for certifying interfaces and will deliver as expected in production environments.

- **Interactive Documentation / Console** - Automatically generated API documentation, which allows developers to make calls against APIs as they're learning about the interface, turning API education into a hands-on experience. Documentation is not just designed for API consumers, they are used as a common way to articulate what an API is, throughout its life – even to non-developer stakeholders.

- **Notebook / Directory** - Providing a local, or cloud-based storage repository, delivering a single place to create and manage API definitions, and execute other API design building blocks. Re-use of common patterns throughout the API design lifecycle is essential in moving the world of APIs forward from the often bespoke world that exists today.

- **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 with API definition. API designers should be able to easily test their interfaces and ensure they meet design, development, management, and other requirements.

- **Debugging** - Manual, automated, and scheduled debugging of API interfaces, providing a detailed look inside API calls, allowing developers to understand problems with API integrations. Debugging at design time ensures that common mistakes are addressed before they can become a QA issue or, even worse, a production matter.

- **Traffic Inspection** - Logging and analysis of API traffic from testing, debugging, and all other API usage during the API design process. As API designers test, debug, and certify their designs, they should be made aware of how the underlying traffic will behave and look like, as close to a production environment as possible.

- **Validator** - Tools for validating API calls, enabling developers to determine which types of calls will be valid, using central API definition and schema. Even across small groups, design standards can be hard to enforce, so validation is an essential part of the design process.

- **Server Code Generators** - Tooling that generates server-side implementations using API definitions in a variety of languages. API designers should be able to go from design to deployment in a single click, something that modern approaches to defining APIs and deploying them via containerized and virtualized infrastructure affords us in 2015.

- **Client-side Code Generator** - Similar to server-side, there should be tooling that generates client-side API code libraries in a variety of languages. There are several open source tools for enabling this functionality, as well as API service providers who provide this as part of the API design process.

- **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. GitHub has quickly risen to play a central role in web and mobile application development, and is a driving force in the containerization movement, so it makes sense that it would be central to API design.

- **Command Line** - Command line tooling for execution of all API building blocks, providing a native interface for developers to execute any aspect of the API design lifecycle. Like APIs, the command-line breaks down barriers between operating systems and platforms.

- **Websockets** - Providing tools for API communication via websockets using API definition as a guide. API design is not always RESTful, and some providers are stepping up to provide API design services, tools, and other resources with a focus on Websockets and other real-time trends in APIs.

- **Translator** - Tools for translating between various API definitions, allowing the transformation from RAML to Swagger, and between each API definition format. API designers should not speak a single API design language, but rather encourage translation and hopefully fluency in multiple API definition formats.

- **Annotation** - The process for designing APIs is similar to other document management, and tools and interfaces for allowing the annotation of API definitions that provides a communication platform centered around the API design process is important, adding a synchronous or asynchronous communication layer that spans the API design lifecycle.

- **Syntax Highlight** - Tools and interfaces for the highlighting of API definitions that provides IDE-like functionality for API designers, making the API design process similar to what developers expect in the development tools they already use. Eventually, all API design editors will have extensible syntax highlighting, which is just as sophisticated as common programming language implementations.

These are just a few of the common building blocks I'm seeing emerge across the API design service providers listed above. There are numerous other features that I know are in the roadmap, or would seemingly make sense to be in here, but I'm waiting until I see clear evidence of them in the space – my objective is to be leading edge, and skeptical bleeding edge, if I can.

Like APIs themselves, these building blocks are the essential components that are quickly defining the evolving API design sector. Following the API philosophy, I hope to see more of these features available as the smallest possible units of value they can be, and also hope they're available as either open APIs or open source tooling. It's essential that API service providers follow the advice they give to the API providers as they deliver services to this space – circle of life and all that.

In my opinion, building blocks are units of value that easily translate from a paid service to an open source tool. Examples of this in action are with the Apigee API studio which is available as part of their suite of online tooling, but you can also download via GitHub, and potentially integrate into your own application. See the connection? (Company == Apigee API Studio) + (Building Block == Editor) + (Tooling = Swagger). This is my formula for making sense of each layer of the API lifecycle.

SOME OF THE OPEN SOURCE TOOLING AND RESOURCES AVAILABLE

After looking for the common building blocks I also work hard to find the open source tools that are being developed and put to use by API designers, and the API service providers that are investing in the space. While I'm looking to build a list of resources that API architects can put to use, I also want to understand where the gaps are – where do we need more API tooling? I'm all for companies making money through selling services to API developers, but the landscape should be equal parts commercial services, and equal parts open tooling if we want the API economy to operate at desired levels.

One of the most important signals a company or organization can send is via their GitHub account, and active repositories and relationships via the platform. This is my primary way of finding the open resources I have listed below. In 2015, GitHub isn't just about code. API providers and service providers are publishing API definitions, design guides, and much more in repositories. I have a lot more tooling that I'd love to publish here, but I'm trying to keep things somewhat orderly and coherent, and taking my time.

Here are the API tools and resources that I've aggregated and published as part of my API design research. As always, if I've missed something let me know.

API Blueprint

- **API Blueprint** (https://github.com/apiaryio/api-blueprint/) - API Blueprint is a documentation-oriented API description language, a couple of semantic assumptions over the plain Markdown. API Blueprint is perfect for designing your Web API and its comprehensive documentation, and also for quick prototyping and collaboration. It's easy to learn and even easier to read – after all, it's 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.

- **Atom Editor API Blueprint Preview** (https://atom.io/packages/api-blueprint-preview) - The Atom Editor API Blueprint preview is a plugin for the Atom editor that allows you to render HTML representation of API Blueprint in the current open Atom editor using CTRL-SHIFT-A. This plugin requires Agilo to be installed and available in your path.

API Definitions

- **API Blueprint** (https://github.com/apiaryio/api-blueprint/) - API Blueprint is a documentation-oriented API description language, a couple of semantic assumptions over the plain Markdown. API Blueprint is perfect for designing your Web API and its comprehensive documentation, and also for quick prototyping and collaboration. It's easy to learn and even easier to read – after all, it's 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 Specification** (http://raml.org) - 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 encouraging 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.

- **Swagger Specification** (https://github.com/swagger-api/swagger-spec/) - Swagger is a simple yet robust representation of a RESTful API, with a large ecosystem of API tooling that includes code generation, interactive documentation, and much more. Currently there are thousands of developers supporting Swagger in almost every modern programming language and deployment environment, using the 100% open source software and specification.

API Design Editor

- **Atom Editor API Blueprint Preview** (https://atom.io/packages/api-blueprint-preview) - The Atom Editor API Blueprint preview is a plugin for the Atom editor that allows you to render HTML representation of API Blueprint in the current open Atom editor using CTRL-SHIFT-A. This plugin requires Agilo to be installed and available in your path.

- **Deployd** (https://github.com/deployd/deployd) - Deployd is the simplest way to build realtime APIs for web and mobile apps. Ready-made, configurable Resources add common functionality to a Deployd backend, which can be further customized with JavaScript Events.

- **RAML API Designer** (https://github.com/mulesoft/api-designer) - API Designer is a standalone/embeddable editor for RAML (RESTful API Modeling Language) written in JavaScript using Angular.JS which, by default, uses an in-browser filesystem stored in HTML5 Localstorage. Mulesoft provides a cloud version of the editor as part of their larger API suite.

- **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 Design Guide

- **18F API Standards** (https://github.com/18f/api-standards) - 18F is a technology team inside the US federal government. 18F is very API-focused: our first project was an API for business opportunities. This document captures 18F's view of API best practices and standards. We aim to incorporate as many of them as possible into our work. APIs, like other web applications, vary greatly in implementation and design, depending on the situation and the problem the application is solving.

- **Heroku - API Design Guide** (https://github.com/interagent/http-api-design) - Heroku has provided their own set of what they call HTTP+JSON API design practices, which I think describes what we do much better than just web API. The guide is designed for internal and external usage, and looks to provide some consistency in API design that anyone can follow.

- **PayPal API Design Standards** (https://github.com/paypal/api-standards/blob/master/api-style-guide.md) - PayPal has developed their own API design standards, providing a common blueprint for their teams to follow, while also transparently sharing with their API community and the wider API industry to follow.

- **Realtime API Design Guide from Fanout** (http://blog.fanout.io/2015/04/02/realtime-api-design-guide/) - An API Design Guide dedicated to helping you understand the common design approaches as well as the pros and cons of realtime API design, showcasing the implementations of 16 public real-time APIs developed by Fanout.io.

- **The RESTed NARWHL** (http://www.narwhl.com/) - NARWHL is a framework intended to provide a roadmap for those needing to implement an API using current best practices but flexible enough to grow into the future. This guide contains a set of API design recommendations you can implement today with the confidence that your API will be RESTful (level 3 according to the Richardson Maturity Model) and able to adapt to future iterations while still making it easier for developers to use.

- **UK Government Service Design Manual for APIs** (https://www.gov.uk/service-manual/making-software/apis.html) - The Martha Lane Fox report called for government to act as a wholesaler, as well as the retail shop front for services and content by mandating the development and opening up of APIs to third parties – this section is a set of guiding principles for exposing a digital service as an API.

- **White House Web API Standards** (https://github.com/whitehouse/api-standards) - This document provides guidelines and examples for White House Web APIs, encouraging consistency, maintainability, and best practices across applications. White House APIs aim to balance a truly RESTful API interface with a positive developer experience (DX).

RAML

- **RAML Specification** (http://raml.org) - 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 encouraging 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.

Swagger

- **Swagger Specification** (https://github.com/swagger-api/swagger-spec/) - Swagger is a simple yet robust representation of a RESTful API, with a large ecosystem of API tooling that includes code generation, interactive documentation, and much more. Currently there are thousands of developers supporting Swagger in almost every modern programming language and deployment environment, using the 100% open source software and specification.

Finding and certifying tooling and other resources that bring value to the space, and verifying each tool is an active project is one of the biggest challenges in this area of my research. I have hundreds of tools targeted for review, but as a one-man show I only have so much time to do the work required. As I get to know each tool and hopefully the people or companies behind them, I will add to this list.

I depend on my readers to help me understand what tools they use, and which ones actually deliver as expected. If you know of an API design tool or resource that you have created, or depend on in your own API design please let me know, or publish to the API design repository. I depend on these conversations with API designers and architects to help me make sense of the volumes of open source tools on GitHub, and also available via other industry sites.

As with other stops along the API lifecycle, the availability of open source tooling is a sign of the health and maturity of the area. I didn't feel that API management was truly a viable business sector until there were enough open tooling resources available when it came to API portals, documentation, analytics, billing, and other key building blocks of API management – and the same is true for API design.

WHY API DESIGN IS IMPORTANT

API design is much more than the naming and ordering of the surface area of any one API. It's about the sharing of design patterns and best practices, and collaborating and communicating with both the technical and business stakeholders of an APIs operation. I don't care how perfectly "designed" your API is from a technical and RESTafarian standpoint – if it doesn't speak to business requirements and also make sense to consumers, it won't matter. In 2015, API design is transcending the technical and impacting every part of business operations.

When technology folks look at API design, they often think of RESTful design principles and other more network, app, and code centered elements of APIs. When business folks get exposed to API design, they think more of the end-business and end-user requirements, taking API design quickly beyond the tech and into the areas that could matter most. A healthy balance between the technology and the business demands of APIs is why the latest API design wave is working, fostering a healthy discussion about the value that APIs can deliver before they are etched in code.

API design is not yet a formal discipline. Early API service providers like Apiary, Mashape, Mulesoft, Restlet, and Apigee have led the charge, but we're at an important turning point. API architects, providers, and designers need to make themselves heard regarding what they're using and, more importantly, what they need for success. As with the early days of API management evolution, we need to cut through the vendor marketing and hype and focus on the services, tooling, and resources that are actually needed – and not just for the sake of selling a service along the API journey.

Another important consideration that makes API design so important at this time is that we now have a handful of viable API definition formats (like API Blueprint, Swagger, RAML, and WADL) available that helps us express and define our digital resources, in a way that we can communicate, share, and collaborate around. The availability and maturity of these formats provide for very rich, fertile environment for API design methodologies, services, and tooling to flourish – if they deliver value. If we do things right, API design can quickly begin to look like the API management space, where we have a number of service providers to choose from, a wealth of open tooling to put to use, and plenty of healthy case studies to showcase how its done right, across multiple industries.

This does not conclude my API design research. As I state in the introduction to this paper, this is just a momentary snapshot of API design, meant to give us an executive understanding of where we stand in July 2015. To help you continue learning until the next update I've included much of the links I've curated, providing the raw research that has gone into my own understanding. Hopefully they'll provide you with more material to go through, until I update this white paper again. If not, remember, all of this is available on GitHub at design.apievangelist.com.

API DESIGN NEWS FROM THE LAST COUPLE OF MONTHS

If you aren't familiar with what I do as the API Evangelist, I've spent the last five years monitoring the API space, curating links from across the web, and putting them into different buckets. This curation then goes into my research, analysis, and ultimately this white paper you're reading.

I've shared just the last 50 links, spanning the 2nd quarter of 2015. A complete list can be found on GitHub at design.apievangelist.com. While most of the links I provide are from 3rd party locations, some of them provide links to my own analysis in the area.

- **The API Design Tooling I Have Included In My Research ·** (06-30-2015 on apievangelist.com) - http://apievangelist.com/2015/06/30/the-api-design-tooling-i-have-included-in-my-research

- **Tightening Up The Organizations That Are Included In My API Design Research** (06-30-2015 on apievangelist.com) -http://apievangelist.com/2015/06/30/tightening-up-the-organizations-that-are-included-in-my-api-design-research

- **API Design Considerations for The Internet of Things** (06-30-2015 on www.programmableweb.com) - http://www.programmableweb.com/news/api-design-considerations-internet-things/analysis/2015/06/30

- **Why It's OK To Design Imperfect APIs** (06-30-2015 on www.programmableweb.com) - http://www.programmableweb.com/news/why-it%e2%80%99s-ok-to-design-imperfect-apis/analysis/2015/06/30

- **The API Design Tooling I Have Included In My Research** (06-29-2015 on apievangelist.com) - http://apievangelist.com/2015/06/30/the-api-design-tooling-i-have-included-in-my-research/

- **Breaking Down Publication References With The Global Change Information System API ·** (06-23-2015 on apievangelist.com) -http://apievangelist.com/2015/06/23/breaking-down-publication-references-with-the-global-change-information-system-api/

- **Lessons learnt from shipping APIs for Microsoft's cloud platform** (06-19-2015 on sriramk.com) - http://sriramk.com/cloud-api-lessons.html

- **Decoupling the Mind of the API Designer** (06-12-2015 on blog.apiary.io) - http://blog.apiary.io/2015/06/09/decoupling-the-mind-of-the-api-designer

- **A Tale of Four API Designs: Dissecting Common API Architectures** (06-11-2015 on nordicapis.com) - http://nordicapis.com/a-tale-of-four-api-designs-dissecting-common-api-architectures/

- **Splitting My Blog API Into Two Separate APIs For News And Analysis ·** (06-10-2015 on apievangelist.com) - http://apievangelist.com/2015/06/10/splitting-my-blog-api-into-two-separate-apis-for-news-and-analysis/

- **Using API Definitions To Help API Providers With Their API Design Roadmap ·** (06-10-2015 on apievangelist.com) -http://apievangelist.com/2015/06/10/using-api-definitions-to-help-api-providers-with-their-api-design-roadmap

- **New JSON API Specification Aims to Speed API Development** (06-10-2015 on www.programmableweb.com) -http://www.programmableweb.com/news/new-json-api-specification-aims-to-speed-api-development/2015/06/10

- **Decoupling the Mind of the API Designer** (06-09-2015 on blog.apiary.io) - http://blog.apiary.io/2015/06/09/decoupling-the-mind-of-the-api-designer/

- **Visions Of My Perfect API Design Editor Using Electron ·** (06-04-2015 on apievangelist.com) - http://apievangelist.com/2015/06/04/visions-of-my-perfect-api-design-editor-using-electron

- **Top 5 Development Tips for a Killer API** (06-04-2015 on nordicapis.com) - http://nordicapis.com/top-5-development-tips-for-a-killer-api/

- **Why Not To Overlook API Planning And What To Do About It** (06-04-2015 on www.programmableweb.com) - http://www.programmableweb.com/news/why-not-to-overlook-api-planning-and-what-to-do-about-it/analysis/2015/06/04

- **Integrations are Hard Part II: API Resources, Search and Pagination** (06-03-2015 on cloud-elements.com) - http://cloud-elements.com/integrations-hard-part-ii-api-resources-search-pagination/

- **Article: The Power of RAML** (06-02-2015 on www.infoq.com) - http://www.infoq.com/articles/power-of-raml

- **How to build APIs efficiently?** (05-30-2015 on api-university.com) - http://api-university.com/blog/how-to-build-apis/

- **Article: From Doodles to Delivery: An API Design Process** (05-26-2015 on www.infoq.com) - http://www.infoq.com/articles/doodles-to-delivery

- **RESTful API Design Part III: Error Handling** (05-26-2015 on cloud-elements.com) - http://cloud-elements.com/restful-api-design-part-iii-error-handling/

- **7 Important API Design Lessons** (05-21-2015 on nordicapis.com) - http://nordicapis.com/7-api-design-lessons-world-tour-roundup/

- **Apigee adds some Swagger to API design** (05-21-2015 on www.pcadvisor.co.uk) - http://www.pcadvisor.co.uk/news/software/apigee-adds-some-swagger-to-api-design-3612750/

- **Apigee API Studio: Designing, Testing and Sharing APIs** (05-21-2015 on www.infoq.com) - http://www.infoq.com/news/2015/05/apigee-api-studio

- **Restlet and SmartBear Partner to Deliver Restlet Studio Plugin for Ready! API** (05-20-2015 on restlet.com) - http://restlet.com/blog/2015/05/20/restlet-and-smartbear-partner-to-deliver-restlet-studio-plugin-for-ready-api/

- **Versioning APIs** (05-15-2015 on blog.clearbit.com) - http://blog.clearbit.com/versioning-apis

- **A Guide to REST and API Design** (05-11-2015 on transform.ca.com) - http://transform.ca.com/rest-api-design-guide.html

- **Do you really know why you prefer REST over RPC?** (05-10-2015 on apihandyman.io) - http://apihandyman.io/do-you-really-know-why-you-prefer-rest-over-rpc/

- **Nearly all web APIs get paging wrong** (05-08-2015 on vermorel.com) - http://vermorel.com/journal/2015/5/8/nearly-all-web-apis-get-paging-wrong.html

- **Guest Post: Why The API Pattern Is Broken And How We Can Fix It ·** (05-05-2015 on apievangelist.com) - http://apievangelist.com/2015/05/05/guest-post-why-the-api-pattern-is-broken-and-how-we-can-fix-it

- **The Role of the API Designer** (05-03-2015 on blog.apiary.io) - http://blog.apiary.io/2015/05/03/the-role-of-the-api-designer

- **Understand about DELETE Verb in Web API RESTful Services using data from both Request Body as well as Uri** (04-28-2015 on www.codeproject.com) -http://www.codeproject.com/tips/986510/understand-about-delete-verb-in-web-api-restful-se

- **api-standards/api-style-guide.md at master · paypal/api-standards · GitHub** (04-24-2015 on github.com) - https://github.com/paypal/api-standards/blob/master/api-style-guide.md

- **The data, the hypermedia, and the documentation** (04-23-2015 on apihandyman.io) - http://apihandyman.io/the-data-the-hypermedia-and-the-documentation/

- **HTTP Verbs Demystified: PATCH, PUT, and POST** (04-21-2015 on cloud-elements.com) - http://cloud-elements.com/http-verbs-demystified-patch-put-and-post/

- **Introduction to Apiary: Overview of Apiary and How to Create APIs** (04-20-2015 on www.developer.com) - http://www.developer.com/design/introduction-to-apiary-overview-of-apiary-and-how-to-create-apis.html

- **REST API design tips** (04-19-2015 on kwtrnka.wordpress.com) - https://kwtrnka.wordpress.com/2015/04/20/rest-api-design-tips/
- **API: Part of the Creative Palette** (04-14-2015 on nordicapis.com) - http://nordicapis.com/apis-part-of-the-creative-palette/
- **Bizcoder - Solving Dropbox's URL Problems** (04-10-2015 on www.bizcoder.com) - http://www.bizcoder.com/solving-dropbox-s-url-problems
- **Solving Dropbox's URL Problems** (04-10-2015 on www.bizcoder.com) - http://www.bizcoder.com:80/solving-dropbox-s-url-problems
- **Here We Go Again, SOA And API — We Have Already Done This! ·** (04-09-2015 on apievangelist.com) - http://apievangelist.com/2015/04/09/here-we-go-again-soa-and-api--we-have-already-done-this/
- **418: I'm a teapot, and other bad API responses** (04-09-2015 on cloud-elements.com) - http://cloud-elements.com/418-im-a-teapot-and-other-bad-api-responses-restful-api-design/
- **Building Reusable REST API Services (Part 3 of 4)** (04-08-2015 on blog.dreamfactory.com) - http://blog.dreamfactory.com/building-reusable-rest-api-services-part-3-of-4
- **In a REST world, there's room for non-REST APIs** (04-07-2015 on www.programmableweb.com) - http://www.programmableweb.com/news/rest-world-there%e2%80%99s-room-non-rest-apis/analysis/2015/04/07
- **JSON API Spec Goes Through The Hacker News Gauntlet** (04-06-2015 on www.programmableweb.com) - http://www.programmableweb.com/news/json-api-spec-goes-through-hacker-news-gauntlet/2015/04/06
- **How to Decide How Many HTTP Status Codes Your API Needs** (04-02-2015 on www.programmableweb.com) -http://www.programmableweb.com/news/how-to-decide-how-many-http-status-codes-your-api-needs/elsewhere-web/2015/04/02
- **InfoQ eMag: Web APIs: From Start to Finish** (03-31-2015 on www.infoq.com) - http://www.infoq.com/minibooks/emag-web-api

- **Bizcoder - API Design Notes: Smart Paging** (03-31-2015 on www.bizcoder.com) - http://www.bizcoder.com/api-design-notes-smart-paging
- **eCommerce API Design – The Good, The Bad, and The Etsy API** (03-30-2015 on cloud-elements.com) - http://cloud-elements.com/ecommerce-api-design-the-etsy-api/
- **The ways of the API smartness** (03-28-2015 on apihandyman.io) - http://apihandyman.io/the-ways-of-the-api-smartness/

Thanks for tuning in to my research!

Remember – you can find all of this on GitHub if you want to get involved!

`{"logo":"API Evangelist"}`

@kinlane

@apievangelist

Make sure and share your public API designs at The API Stack – otherwise all of this won't work!