{“API”:”Definition”}

Version 2017.03 | API Specifications, Schema, and Scopes For The API Economy

OpenAPI Spec 3.0

The machine-readable OpenAPI Spec is approaching version 3.0 after entering into the Linux Foundation as part of the Open API Iniative (OA)--the first major major release executed by the governing organization. I wanted to take a look at the highlights of this release, and what might be next for the API specification.

Human Services

API definitions are not just for startups. The Human Services Data Specification (HSDS) is now defined also as an OpenAPI, providing a machine readable blueprint that any city, county, state, or federal agency can use to describe their vital human services-- learn more about the API definition as well as the data schema.

INDEX SECTION

Table of Contents

04

Definitions

Exploring the leading formats when it comes to API definitions, schema, and stardards. Including a look at the leading API specification formats, the most common media types, as well as a peek at the hymerdia media types being put to use across the API space.

05

Services

There are three distinct leaders of services when it comes to API definitions. All three of the leading providers have a horse in the race when it comes to API specications. In 2017, will we see all three of them join together as part of the Open API specification group?

06

Tooling

A quick look at some of the tooling available when it comes to working with API definitions, schema, and scopes. In this edition I a focusing on generating, parsing, and converting API definitions into a variety of application formats that can be used anywhere.

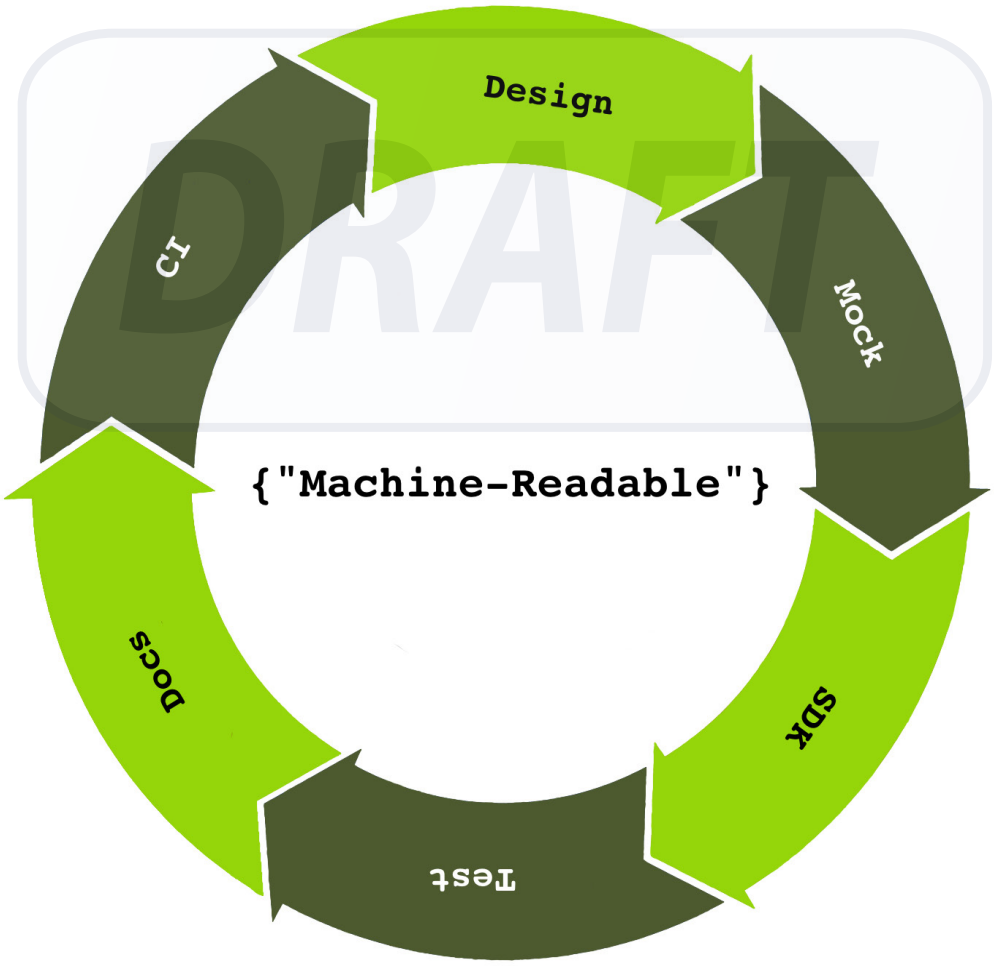
EDTRL

API Definition is an API Evangelist production, operated by Kin Lane. It is a solo effort from someone who has been monitoring the technology, business, and politics of APIs since 2010.

This guide is designed to be a summary of the world of API definitions, providing the reader with a recent summary of the variety of specifications that are defining the technology behind almost every industry, driving the web, mobile, and device applications touch almost every part of our digital worlds.

API Definition

Industry Guide



Definitions are used at every stop along the API life cycle today.

API definitions, schema, and scopes are playing a centrol role across every industry -- providing a common language that can be used across web, mobile, device, and other applications.

WRITTEN BY: KIN LANE

In the last five years, a handful of new specification formats have emerged, allowing us to better describe what application programming interfaces (API) do, and how they can be put to work across the software development life cycle. These new API definition formats allow for the location, response, requests, and other essential technical, business, and legal aspects of what an API does to be described in a human readable (YAML), but also machine-readable (JSON) way -- fueling new ways to think about digital automation on the web.

In 2012, it was common for API definitions to be developed to assist in the generation and maintenance of API documentation--keeping the programmatic interface in sync with the human (consumer) interface. As the API sector continued to evolve, these same API definitions started to become applied as part of the API design process, allowing key technical and business stakeholders to communicate around the design of the API, even before any code was written, and documentation is even needed. This changed how we develop APIs, allowing stakeholders to be involved much earlier on in the development process, and saving valuable development cycles.

In 2016, these API definitions are being used to design, mock, deploy, document, manage, test, generate SDKs, and numerous others stops along the API life cycle. Additionally, evolution in defining the underlying data model used in API requests and responses have also been pushed forward, further refining how we define and communicate about OAuth scopes that govern identity and access to APIs, and the valuable data, content, and algorithms they expose.

API Definition is a regular snapshot of this layer of the API space that feeds and drives every other aspect of the API life cycle. API definitions will continue to play an important role for API providers, as well as API consumers, and are defining the next wave(s) of services that target the sector. API Definition will be covering the emergence of industry specific standards such as PSD2 governing the banking sector, and Open Referral focused on delivering human services.

This is your source for summary of the world of API efinitions, providing some of the latest changes in the space, and the standards that are being baked into how we are doing business via the web, mobile, and device applications that APIs are driving today.

Open API Specification 3.0

01

The API specification has reached version 3.0 -- what are the details, and what does 2017 hold for the specification?

November, 5th 2015

The Open API Iniative

Entered the Linux Foundation

WRITTEN BY: KIN LANE

The OpenAPI Spec, formerly known as Swagger is approaching an important milestone, version 3.0 of the specification, but it is also the first major release since the specification was entered into the Linux Foundation. Swagger was the creation of Tony Tam of Wordnik back in 2010 , but after the project was acquired by SmartBear in 2015, the company donated the specification to the newly formed OpenAPI Initiative (OAI) which is part of the Linux Foundation. Swagger is now been reborn as the OpenAPI Specification, and in early 2017 is approaching it's first major relase under the guidance of a diverse group of governing members.

Version 3.0 of the API specification format has taken a much more modular, and reusable approach to defining the surface area of an API, enabling more power and versatility when it comes to describing the request and response models, as well as providing details on the common components that make up API usage like the underlying data schema and security definitions. There are numerous changes to the API specification, but there are just a handful that will have a significant impact across every stop along the API life cycle where API definitions are making an impact..

The most notable shift is the componentized, modular, reusable aspect of the specification--after that, it is the content negotiation, linking, and web hooks.

Hosts

When describing your API, you can now provide multiple hosts, allowing you to better deal with the complexity of how APIs might be located in a single location, or spread across multiple cloud location, and global infrastructure.

Content Negotiation
You can now provide content objects to define the relationship between response objects, media types, and schema, opening up the negotiation of exactly the type of content needed, encouraging the design of multiple rich dimensions for any single API resource.

Body
The latest version of the specification plays catch-up when it comes to allowing the body of a request and response to be defined separately from the request parameters, allowing for more control over the payload any API calls.

Schema
There is an increased investment in JSON Schema, including the support of `oneOf`, `anyOf` and `not` functions, allowing for alternative schema, as well as the standard JSON schema definition included.

Components
The new components architecture really reflects “APIs” in my opinion, making everything very modular, reusable, and much more coherent and functional. The new version encourages good schema and component reuse, helping further bringing the definition into focus.

Linking
White not quite full hypermedia, version 3.0 of the OpenAPI Spec supports linking, allowing for the description of relationships between paths, giving a nod towards hypermedia design patter, making this version the most resilient so far.

Webhooks
Like the nod towards hypermedia, the specification now allows for the inclusion of callbacks that can be attached to a subscription operation describing an outbound operation--providing much needed webhook descriptions as part of API operations, making it much more real time and event driven.

Examples
The new version enables API architects to better describe, and provide examples of APIs responses and requests, helping make API integration a learning experience, by providing examples for use beyond just documentation descriptions.

Cookies
Responding to a large number of API providers, and the reality on the ground for many implementations, the new version allows for the defining of cookies as part of API requests.

These nine areas represent the most significant changes to the OpenAPI Spec 3.0. The most notable shift is the componentized, modular, reusable aspect of the specification. After that, it is the content negotiation, linking, web hooks, and other changes that moving the needle. It is clear that the 3.0 version of the specification has considered the design patterns across a large number of implementations, providing a pretty wide reaching specification for defining what an API does, in a world where every API can be a special snowflake.

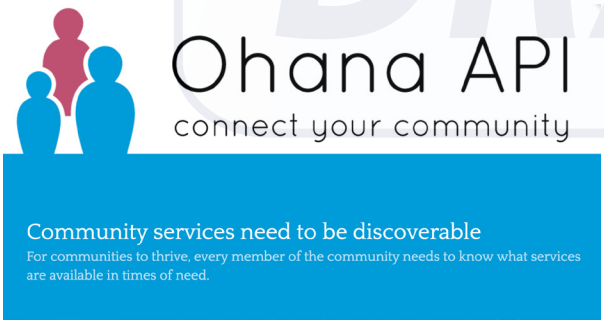
In 2017, the OpenAPI Spec is the clear leader of the API definition formats, with the largest adoption, as well as the amount of tooling developed. While documentation and SDK generation are still the top two reasons for crafting API definitions, there are numerous other reasons for using API definitions including mocking, testing, monitoring, IDE integration, and much, much more. With the introduction of version 3.0 of the OpenAPI Spec in early 2017, the rest of the year will surely be about playing catch up with many of the existing service and tooling providers who speak OpenAPI Spec, introducing the benefits of the new specification into their API solutions.

“2017 will bring a new generation of services and tooling for API providers who support the OpenAPI Spec.”

Human Services

02

API definitions aren't just for tech startups, our cities are depending on them for delivering human services in our communities.



WRITTEN BY: KIN LANE

A lot of attention is given to APIs and the world of startups, but in 2017 this landscape is quickly shifting beyond just the heart of the tech space, with companies, organizations, institutions, and government agencies of all shapes and sizes are putting APIs to work. API definitions are being applied to the fundamental building blocks of the tech sector, quantifying the compute, storage, images, videos, and other essential resources powering web, mobile, and device based applications. This success is now spreading to other sectors, defining other vital resources that are making a real impact in our communities.

One API making an impact in communities is the Human Services Data Specification (HSDS), also known as the Open Referral Ohana API. The project began as a Code for America project, providing an API, website, and administrative system for managing the organizations, locations, and the human services that communities depend on. Open Referral, the governing organization for HSDS, and the Ohana API is working with API Evangelist and other partners to define the next generation of the human services data specification, API definition, as well as the next generation of API, website, admin, and developer portal implementations.

The HSDS API isn't about any single API, it is a suite of API-first definitions, schema, and open tooling that any organization can download or employ to manage vital

The HSDS Specification API isn't about any single API, it is a suite of API-first definitions, schema, and open tooling that cities, counties, states, and federal government agencies can download or fork, and employ to help manage vital human

services for their communities. Providing not just a website for finding vital services, but a complete API ecosystem that can be deployed incentivizing developers to build important web, mobile, and other applications on top of a central human services system. Better delivering on the mission of human services organizations, and meeting the demands of their constituents.

This type approach to delivering APIs centers around a common data schema, extending it as an OpenAPI Spec definition, describing how that data is accessed, and put to use across a variety of applications, including a central website and administrative system. While server-side HSDS API implementations, website, mobile, administrative, developer portal, and other implementations are important, the key to the success of this model is a central OpenAPI definition of the HSDS API. This definition connects all the implementations within an API's ecosystem, but it also provides the groundwork for a future where all human services implementations are open and interoperable with other implementations--establishing a federated network of services meeting the needs of the communities they serve.

Right now each city is managing one or multiple human service implementations. Even though some of these implementations operate in overlapping communities, few of them are providing 3rd party access, let alone providing integration between overlapping geographic regions. The HSDS API approach employs an API-first approach, focusing on the availability and access of the HSDS schema, then adding on a website, administrative and API developer portals to support. This model opens up human services to humans via the website, which is integrated with the central API, but then also opens up the human services for inclusion into other websites, mobile and device applications, as well as integration with other systems.

The HSDS OpenAPI spec and schema provide a reusable blueprint that can be used to standardize how we provide human services. The open source approach to delivering definitions, schema, and code reduces the cost of deployment and operation for cash strapped public organizaitons and agencies. The API-first approach to delivering human services also opens up resources for inclusion in our applications and system, potentially outsourcing the heavy lifting to trusted partners, and 3rd party developers interested in helping augment and extend the mission of human service organizations and groups.

If you'd like to learn more about the HSDS API you can visit Open Referral (http://openreferral.org). From there you can get involved in the discussion, and find existing open source definitions, schema, and code for putting HSDS to work. If you'd like to contribute to the project, there are numerous opportunities to join the discussion about next generation of the schema and OpenAPI Spec, as well as develop server-side, and client-side implementations. Open Referral can also help you find local organizations that might need assistance on the ground when it comes to delivering human services on the ground within communities.

Open Referral began as part of the DC Open211 project, and has been co-sponsored by Code for America (CfA), in partnership with the Ohana project. It is now a community of practice with multiple pilot projects across the world.

WebConcepts.info

03

The Web's Uniform Interface is based on a large & growing set of specifications--establishing a shared concepts across providers & consumers.

Web Concepts helps soften, and make these critical concepts and specifications accessible and digestible for a new generation of web and API developers.

WRITTEN BY: KIN LANE

Keeping up with the standards bodies like International Organization for Standardization (ISO) and Internet Engineering Task Force (IETF) can be a full time job. Thankfully, Erik Wilde (@dret) has help simplified and made the concepts and specifications that make the web work more accessible and easier to understand, with his WebConcepts.info project.

According to Erik, "the Web's Uniform Interface is based on a large and growing set of specifications. These specifications establish the shared concepts that providers and consumers of Web services can rely on. Web Concepts is providing an overview of these concepts and of the specifications defining them." His work is a natural fit for what I am trying to accomplish with my API definition industry guide, as well as supporting other areas of my research.

One of the areas that slows API adoption is a lack of awarness of the concepts and specifications that make the web work among developers who are providing and consuming APIs. The modern API leveragegs the same technology that drives the web--this is why it is working so well. The web is delivering HTML for humans, and APIs are using the same to deliver machine readable data, content, and access to algorithms online. If a developer is not famiilar with the fundamental building blocks of the web, the APIs they provide, and the applications they build on top of APIs will always be deficient.

The Web Concepts project provides an overview of 28 web concepts with 643 distinct implementations aggregated across five separate organizations including the International Organization for Standardization (ISO), Internet Engineering Task Force (IETF), Java Community Process (JCP), Organization for the Advancement of Structured Information Standards (OASIS), and the World Wide Web Consortium (W3C)--who all contribute to what we know as the web. An awareness and literacy around the 28 concepts aggregated by Web Concepts is essential for any API developer and architect looking to fully leverage the power of the web as part of their API work.

After aggregating the 28 web concepts from the five standards organization, Web Concepts additionally aggregates 218 actual specifications that API developers, architects, and consumers should be considering when putting APIs to work. Some of these specifications are included as part of this API Definition guide, and I will be working to to add additional specifications in future editions of this guide, as it makes sense. The goal of this guide is to help bring awarness, literacy, and profiency with common API and data patterns, making use of the work Web Concepts project, and building on the web literacy work already delivered by Erik, just makes sense.

Web Concepts is published as a Github repository, leveraging Github Pages for the website. He has worked hard to make the concepts and specification available as JSON feeds, providing a machine-readable feed that can be integrated into existing API design, deployment, and management applications--providing web literacy concepts and specifications throughout the API life cycle. All JSON data is generated from the source data, which is managed as a set of XML descriptions of specifications, with the build process based upon XSLT and Jekyll, providing multiple ways to approach all concepts and specifications, while maintaining the relationship and structure of all the moving parts that make up the web.

When it comes to the startup space, the concepts that make up the web, and the specifications that make it all work, might seem a little boring, something only the older engineers pay attention to. Web Concepts helps soften, and make these critical concepts and specifications accessible and digestible for a new generation of web and API developers--think of them as gummy versions of vitamins. If we are going to standardize how APIs are designed, deployed, and managed--making all of this much more usable, scalable, and interoperable, we are going to have to all get on the same page (aka the web).

Web Concepts is an open source project, and Erik encourages feedback on the concepts, and specifications. I encourage you to spend time on the site regularly, and see where you can integrate the JSON feeds into your systems, services, and tooling. We have a lot of work ahead of use to make sure the next generation of programmers have the base amount of web literacy necessary to keep the web strong and healthy. There are two main ways to contribute to the building blocks of the web, participate as a contributor to the standards body, or you can make sure you are implementing common concepts and specifications throughout your work, contributing to the web, and not just walled gardens and closed platforms.

Many API architects are deficient in an awareness of the concepts and specifications that make the web work.

API Specifications

04

The leading specification formats for describing the response, request & security model of web APIs--defining what it does.

OpenAPI Specification

The goal of The OpenAPI Specification 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.

URL: http://apis.how/openapi-spec/

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 but also for quick prototyping and collaboration. .

URL: http://apis.how/api-blueprint/

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.

URL: http://apis.how/raml/

Postman Collection

A collection groups individual requests together., allowing them be organized into folders, accurately mirrorring your API. Requests can also store sample responses when saved in a collection. You can add metadata like name & description too so that all the information that a developer needs to use your API is available.

URL: http://apis.how/pstmn-col/

RESTful API Description Language (RADL)

RESTful API Description Language (RADL) is an XML vocabulary for describing Hypermedia-driven RESTful APIs. Unlike most HTTP API description languages, RADL focuses on defining a truly hypermedia-driven REST API from the clients point of view, moving the conversation forward.

URL: http://apis.how/radl/

RESTful Service Description Language (RSDL)

The RESTful Service Description Language (RSDL) is a machine- and human-readable XML description of HTTP-based web applications (typically REST web services), adding another way to describe APIs in a machine readable format.

URL: http://apis.how/rsdl/

OPEN API INITIATIVE

The Open API Initiative (OAI) was created by a consortium of forward-looking industry experts who recognize the immense value of standardizing on how REST APIs are described. As an open governance structure under the Linux Foundation, the OAI is focused on creating, evolving and promoting a vendor neutral description format. While the initiative focuses on the OpenAPI Specification, other format providers like Apiary are also part of the iniative opening up the potential for the effort to impact beyond just a single API definition.

Data Schema

04

Formats for defining data schema, fields, default values, and other elements for use in all API requests & responses.

JSON Schema

Describes your JSON data format in clear, human- and machine-readable documentation that is complete structural validation, useful for automated testing, and validating client-submitted data.

URL: http://apis.how/json-schema/

Application-Level Profile Semantics (ALPS)

A data format for defining application-level semantics, similar to HTML microformats. ALPS can be used as a profile to explain the application semantics agnostic media types.

URL: http://apis.how/alps/

Asset Description Metadata Schema (ADMS)

ADMS is a profile of DCAT used to describe semantic assets as highly reusable metadata (e.g. xml schemata, generic data models) and reference data (e.g. code lists, taxonomies, dictionaries, vocabularies) for system development

URL: http://apis.how/adms/

Markdown Syntax for Object Notation (MSON)

MSON is a plain-text, human and machine readable, description format for describing data structures in common markup formats such as JSON, XML or YAML.

URL: http://apis.how/mson/

Open Data Protocol (OData)

OData (Open Data Protocol) is an OASIS standard that defines the best practice for building and consuming RESTful APIs. OData helps you focus on your business logic while building RESTful APIs without having to worry about the

URL: http://apis.how/odata/

Docson

Docson is focused more on just documentation, giving your JSON types, using a JSON schema to generate a beautiful documentation, but in a way that could be used for other purposes similiar to what other schemas can do.

URL: http://apis.how/docson/

schema.org

Thing > Organization > LocalBusiness

Schema.org

Schema.org is a collaborative, community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond. Schema.org vocabulary can be used with many different encodings, including RDFa, Microdata and JSON-LD.

Common Media Types

04

The commonly used API media types as defined by IANA, providing a base set of content types that all APIs should be serving up.

text/html

Provide HTML media types for API requests and responses.

application/xml

Provide XML media types for API requests and responses.

application/atom+xml

Provide ATOM media types for API requests and responses.

application/json

Provide JSON media types for API requests and responses.

application/csv

Provide comma separate value (CSV) media types for API requests and responses.

text/tab-separated-values

Provide tab separated values (TSV) media types for API requests and responses

APIs should offer a range of media types allowing consumers to negoiate the best content type for their needs

Internet Assigned Numbers Authority

iana

The Internet Assigned Numbers Authority (IANA) is a department of ICANN, a nonprofit private American corporation that oversees global IP address allocation, autonomous system number allocation, root zone management in the Domain Name System (DNS), media types, and other Internet Protocol-related symbols and Internet numbers.

URL: http://apis.how/iana/

Hypermedia Media Types

04

The leading hypermedia formats being applied during the API design process providing a much more rich API integration experience.

JSON API

JSON API is a specification for how a client should request that resources be fetched or modified, and how a server should respond to those requests.

URL: http://apis.how/json-api/

Collection+JSON

Collection+JSON is a JSON-based read/write hypermedia-type designed to support management and querying of simple collections.

URL: http://apis.how/coll-json/

Siren

Siren is a hypermedia specification for representing entities. As HTML is used for visually representing documents on a Web site, Siren is a specification for presenting entities via a Web API.

URL: http://apis.how/siren/

JSON-LD

JSON-LD, or JavaScript Object Notation for Linked Data, is a method of encoding Linked Data using JSON. It was a goal to require as little effort as possible from developers to transform their existing JSON to JSON-LD.

URL: http://apis.how/json-ld/

Hypertext Application Language (HAL)

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

URL: http://apis.how/hal/

Mason

Mason is a JSON format for introducing hypermedia elements to classic JSON data representations.

URL: http://apis.how/mason/

Foxy.io

The Foxy Hypermedia API is designed to give you complete control over all aspects of your Foxy accounts, whether working with a single store or automating the provisioning of thousands. Anything you can do within the Foxy administration, you can also do through the API.

URL: http://apis.how/foxy-api/

U.S. Data Federation

Aggregate API specifications across government

The U.S. Data Federation is a federal government effort to facilitate data interoperability and harmonization across federal, state, and local government agencies by highlighting common data formats, API specifications, and metadata vocabularies. The project is focusing on being a coordinating interoperability across government agencies by showcasing and supporting use cases that demonstrate unified and coherent data architectures across disparate agencies, institutions, and organizations.

The project is designed to shine a light on "emerging data standards and API initiatives across all levels of government, convey the level of maturity for each effort, and facilitate greater participation by government agencies"--definitely in alignment with the goal of this guide. There are currently seven projects profiled as part of the U.S. Data Federation, including Building & Land Development Specification, National Information Exchange Model, Open Referral, Open311, Project Open Data, Schema.org, and the Voting Information Project.

By providing a single location for agencies to find common schema documentation tools, schema validation tools, and automated data aggregation and normalization capabilities, the project is hoping to incentivize and stimulate reusability and interoperability across public data and API implementations. Government agencies of all shapes and sizes can use the common blueprints available in the U.S. Data Federation to reduce costs, speed up the implementation of projects, while also opening them up for augmenting and extending using their APIs, and common schema.

It is unclear what resources the U.S. Data Federation will have available in the current administration, but it looks like the project is just getting going, and intends to add more specifications as they are identified. The model reflects an approach that should be federated and evangelized at all levels of government, but also provides a blueprint that could be applied in other sectors like healthcare, education, and beyond. Aggregating common data formats, API specifications, metadata vocabularies, and authentication scopes will prove to be critical to the success of the overall climate of almost any industry doing business on the web in 2017.

Current Projects

Building & Land Development

Information Exchange Model

Open211

Open311

Project Open Data

Schema.org

Voting Information Project.

"A model that other sectors should follow to harmonize across the industry"

API Transformer

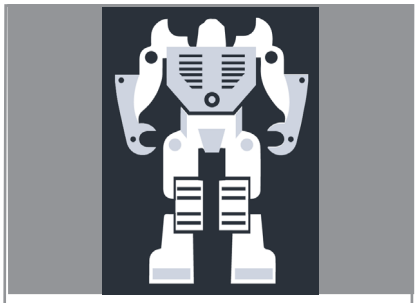
Convert API definitions into multiple formats

OpenAPI Spec is currently the most used API definition format out there currently, with the number of implementations, and tooling, with API Blueprint, Postman Collections, and other formats trailing behind. It can make sense to support a single API definition when it comes to an individual platforms operations, but when it comes to interoperability with other systems it is important to be multi-lingual and support multiple of the top machine-readable formats out there today.

In my monitoring of the API sector, one service provider has stood out when it comes to being a truly multi-lingual API definition service provider--the SDK generation provider, APIMATIC. The company made API definitions the heart of it's operations, generating what they call development experience (DX) kits, from a central API definition uploaded by users--supporting OpenAPI Spec, API Blueprint, Postman Collections, and other top formats. The approach has allowed the company to quickly expand into new areas like documentation, testing, continuous integration, as well as opening up their API definition translation as a separate service called API Transformer.

API Transformer allows anyone to input a API Blueprint, Swagger 1.0 - 1.2, Swagger 2.0 JSON, Swagger 2.0 YAML, WADL - W3C 2009, WSDL - W3C, Google Discovery, RAML 0.8 - 1.0, I/O Docs - Mashery, HAR 1.2, Postman Collection 1.0 - 2.0, Mashape, or APIMATIC Format API definition and then translate and export in a API Blueprint, Swagger 1.0 - 1.2, Swagger 2.0 JSON, Swagger 2.0 YAML, WADL - W3C 2009, RAML 0.8 - 1.0, Postman Collection 1.0 - 2.0, and their own APIMATIC Format. You can execute API definition translations through their interface or seamlessly integrate with the API Transformer API definition conversation API.

There is no reason that an API provider and API service providers shouldn't be multi-lingual. It is fine to adopt a single API definition as part of your own API operations, but when it comes to working with external groups, there is no excuse for not being able to work with any of the top API definition formats. The translation of API definitions will increasingly be essential to doing business throughout the API life cycle, requiring each company to have an API definition translation engine baked into their continuous integration work flow, transforming how they do business and build software.



Formats Supported

API Blueprint

Swagger 1.0 - 1.2

Swagger 2.0 JSON

Swagger 2.0 YAML

WADL - W3C 2009

RAML 0.8 - 1.0

Postman Collection 1.0 - 2.0

APIMATIC Format

"Being multi-lingual when it comes to API definitions is essential"

API Services

05

The companies who are offering commercial services behind the API definition formats they have invested in as open standard.

Swagger

Swagger is a powerful open source framework backed by a large ecosystem of tools that helps you design, build, document, and consume your RESTful APIs. The Swagger framework is supported by a set of core tools for designing, building, and documenting RESTful APIs. All of these tools are free and open source projects available on GitHub.

URL: http://apis.how/swagger/

Apiary

Apiary.io is a suite of tools that help companies build web APIs, test, monitor, and document them. 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.

URL: http://apis.how/apiary/

Mulesoft

MuleSoft is a B2B application delivery network that connects applications, data, and devices with APIs. It enables organizations to improve their applications through integration, without the need for custom point-to-point code. Mulesoft also provides instant API connectivity to hundreds of on-premise and cloud-based applications and systems.

URL: http://apis.how/mulesoft/

Each of these providers have a horse in the API definition race!

3Scale

3scale provides Free & Enterprise API management solutions for developers, and companies to securely distribute, control, manage, operate and monetize their API to 3rd parties (e.g. developers, business partners, etc). 3scale API management platform is Plug and Play, configurable, flexible and scalable and brings API providers an unparalleled level of ownership, control and configuration to ensure QoS, SLA and a great user experience.

URL: http://apis.how/3scale/

Tooling - Generators

06

Some of the tooling for generating API definitions

Swagger Maven Plugin

This plugin enables your Swagger-annotated project to generate Swagger specs and customizable, templated static documents during the maven build phase.

URL: http://apis.how/swagger-maven/

Grunt Swagger Generator

A Grunt solution for building OpenAPI Speicification using a JavaScript task runner, a tool used to automatically perform frequently used tasks such as minification, compilation, unit testing, linting, etc.

URL: http://apis.how/grunt-swagger/

Swagger.json Generation

The toolkit has a command that will let you generate a swagger spec document from your code. The command integrates with go doc comments, and makes use of structs when it needs to know of types.

URL: http://apis.how/swagger-json/

Python APISpec

A pluggable API documentation generator. Currently supports the generation of Swagger 2.0 specification for use in Python applications, and can be used beyond just API documention it is intended for.

URL: http://apis.how/python-apispec/

Paw API Blueprint Generator Extension

Paw is a full-featured HTTP client that lets you test the APIs you build or consume, and the Paw extension providing support to export API Blueprint as a code generator.

URL: http://apis.how/paw-generator/

Swagger-PHP

Generate interactive Swagger documentation for your RESTful API using doctrine annotations. Providing a solution for generating API definitions in a PHP environment.

URL: http://apis.how/swagger-php/

Restlet

Restlet

The Restlet API Platform enables developers and non-developers to design, create, run and manage the APIs that provide access to any data or application. Restlet Framework is the most widely used open source solution for Java developers who want to create and use APIs. The first Platform-as-a-Service dedicated to web APIs, APISpark enables any organization to become an API provider in minutes via an intuitive browser interface.

URL: http://apis.how/restlet/

Tooling - Parsers

06

Some of the tooling for parsing API defini-

Swagger Parser (JS)

Parses, validates, and dereferences JSON/YAML Swagger specs in Node and browsers. Parses Swagger specs in JSON or YAML format Validates against the Swagger 2.0 schema and the Swagger 2.0 spec

URL: http://apis.how/swagger-parser-js/

Prance (Python)

Prances provices parsers for Swagger/OpenAPI 2.0 API specifications in Python. It uses swagger_spec_validator to validate specifications, but additionally resolves JSON references in accordance with the Swagger spec.

URL: http://apis.how/prance-python/

Swagger Parser (PHP)

Parses Swagger specification documents programmatically, and also parses data structures according to a specification in PHP.

URL: http://apis.how/swaggr-parser-php/

Swagger Parser (Java)

A swagger parser project, which reads OpenAPI Specifications into current Java POJOs. It also provides a framework to add additional converters into the Swagger objects, making the entire toolchain available.

URL: http://apis.how/swaggr-parser-java/

Swagger Parser (Ruby)

Swagger is a Ruby library for parsing, building, and traversing Swagger API definitions.

URL: http://apis.how/swagr-parser-ruby/

Snow Crash

Snow Crash is the reference API Blueprint parser built on top of the Sundown Markdown parser. API Blueprint is Web API documentation language.

URL: http://apis.how/snow-crash/

Tyk

An open source, lightweight, fast and scalable API Gateway. Set rate limiting, request throttling, and auto-renewing request quotas to manage how your users access your API. Tyk supports access tokens, basic auth & OAuth 2.0 to integrate old and new services. Tyk can record and store detailed analytics which can be segmented by user, error, endpoint and client ID across multiple APIs and versions.

URL: http://apis.how/tyk/

Tooling - Converters

06

Some of the tooling for parsing API definitions

Swagger2Blueprint

Convert Swagger API descriptions into API Blueprint

URL: http://apis.how/swagger2blueprint/

Swagger2Postman

Creates a Postman collection from live Swagger documentation

URL: http://apis.how/swagger2postman/

Swagger2Markup (Java)

Swagger2Markup converts a Swagger JSON or YAML file into AsciiDoc or Markdown documents which can be combined with hand-written documentation.

URL: http://apis.how/swagger2markup/

API Spec Converter (Node.js)

A tool for converting from other API specification formats (e.g. I/O Docs and API Blueprint) to Swagger

URL: http://apis.how/apispec-converter/

RAML Converter

Library to convert RAML files to other formats+

URL: http://apis.how/raml-converter/

LucyBot API Spec Converter (JavaScript)

Convert API descriptions between popular formats such as OpenAPI(fka Swagger), RAML, API Blueprint, WADL, etc.

URL: http://apis.how/lucybot-converter/

Dreamfactory

DreamFactory is a free, open source software package that provides a complete REST API for mobile, web, and IoT applications. Easy to use installation packages are available for your server, cloud, and desktop computer. Hook up any SQL or NoSQL database, file storage system, or external service and DreamFactory instantly creates a comprehensive REST API platform with live documentation, user management, and more..

URL: http://apis.how/dreamfactory/

About The Sponsors

Those Who Make This Public Possible - Thanks!

This publication is targeting API providers at startups, companies, organizations, institutions, and government agencies, providing a snapshot of the API sector specifically dedicated to API definitions. While the primary target of the publication is API providers, there is also regular updates about best practices, new tooling, standards, and other aspects of the sector that concern API service providers, and companies looking to reach API providers in the space.

It is these API service providers that support this publication. These companies fund the publication of this guide through the sponsorship of feature and short posts, as well as other designated slots throughout the publication.

These are the companies supporting this edition of API definition:

- 3Scale - http://apis.how/3scale
- Restlet - http://apis.how/restlet
- Dreamfactory - http://apis.how/dreamfactory
- Tyk - - http://apis.how/tyk

If you would like to find out about upcoming publishing opportunities, and supporting this research, you can email info@apievangelist.com to learn more. This guide is usually updated monthly with a regular rotation of feature articles, and shorts, as well as including new specifications, services, tooling, and other relevant elements from across the API sector--new opportunities emerge on a regular basis.

The primary objective of this publication is to reach and educate API providers, while also including API service providers in this regular storytelling. The API Evangelist website provides access to the raw research on API definitions, but this guide is meant to provide an executive summary that any business or technical user can pick up to get up to speed on the subject.

Email info@apievangelist.com to become a sponsor of API Definition today.

SPONSOR

Become Sponsor

Email info@apievangelist to get more information about sponsoring this publication.

Striking Balance

API Evangelist sponsors make it so I can dedicate my time to researching the API space, as well as provide these guides to what is going on -- I couldn't do it without them.

"Thank You!"

About the Publication

Brought To You By Kin Lane, the API Evangelist

I have been researching the API sector for the last seven years. API Evangelist is the real time result of this research, providing analysis, links to news, patents, companies, and the APIs that are making an impact on the space. Since 2012 I have worked to publish white papers, guides, and other more long form content derived from my research -- this is the latest attempt to provide access to my API definition research in a more polished and portable format.

As I monitor the space I keep an eye on what companies, organizations, institutions, agencies, and individuals are up to, aggregating best practices, services, tooling, specifications, and the other building blocks that make this API thing work. While there is a wealth of raw research available publicly on my site, my readers have continued to request a more refined, polished, distilled down version of my research, resulting in the delivery of these industry guides.

With this latest release of my API definition industry guide, I'm bringing more design to the table, making the guide look better, but it is also something that is forcing me to think through the material I provide a little deeper than I did before. Earlier editions of this guide were quantity over quality, and with this next wave of guides I'm flipping that over, and emphasizing analysis, storytelling, and smaller, more refined amounts of content--focusing on helping people make sense of things, and hopefully not overwhelming them with information.

Along the way, I am also experimenting with new ways of generating revenue and paying the bills. I have a number of companies asking to sponsor my site, but alas I only have four logo slots available. These polished versions of my guides give me a new way to extend sponsorship opportunities to my partners, and companies doing interesting things in the space. The money helps me dedicate my time to keeping tabs on what is going on and generating as high-quality content as possible.

Thank you for tuning in, and please let me know how I can make this publication better -- thanks for your support.

-- Kin Lane

ABOUT

Kin Lane

twitter: @kinlane

github: @kinlane

kinlane.com

API Evangelist

twitter: @apievangelist

github: @apievangelist

apievangelist.com

a

{“API”:”Evangelist”}

{“API”:”Definition”}

API Specification, Schema, and Scopes For The API Sector

Providing a snapshot of the world of API definitions, with details on the API specifications, schema, scopes, and other building blocks of the API sector -- being used across almost every stop along the API life cycle.