



Documentation: APIs, Usability and Tool Chain

Greg Glover – Documentation PTL

February 8, 2018

ONAP Documentation – Beijing Scope

ONAP Wiki

Informal, Open Community-based

- **Introducing ONAP Platform**
 - High level Flows & Use Cases
- **ONAP “light”**
 - SE’s/Architects try out platform
- **Wiki Sandbox**
 - Developers try out code
- **Forums**
 - Feedback
 - Discussion
- **Community**

New
Doc
Projects

Readthedocs

Formal, Change Controlled by Release

- **Release Notes**
- **Developer Guides**
- **User Guides**
- **Modeling & Arch**
- **API Doc & Guidelines**
- **Glossary / Reference**
- **Enhanced Tool Chain**

Beijing
Updates

New Doc
Projects

Migrate
seed
content

ONAP API Documentation – Agenda

1. Amsterdam Status

2. Beijing Goals

- Format & Structure
- ReadtheDocs
- Versioning

3. Community / TSC Questions

- SwaggerHub
- Model enabled vs. API driven
- Hosting

API Sub-Team Members

Rich Bennett - AT&T
Dana Bobko – AT&T
Tara Cummings - Erickson
Andy Meyer – AT&T
Dave Neary – Red Hat
Jack Pugaczewski – CenturyLink
Rene´ Robert – Orange

Other Doc Team Members

Eric Debeau´ - Orange
Audrey Fry – Accenture
Thanh Ha – LF
Anuj Kapoor – Amdocs
Victor Morales – Intel
Pawel Pawlak – Orange
Steven Wright – AT&T



ONAP API Documentation – Amsterdam Status

1. Good base

- APIs generally well-documented across projects (**thank you PTLs!**)
- Most projects use some sort of model in their software (e.g. SDNC/APPC using ODL MD-SAL). A&AI and graphical data base schema, probably some Django web apps in other places
- Most projects have export programs and/or have a spec in process

2. Areas to improve

- Format – no standard list of attributes and definitions, etc.
- Repository – API documentation both in ReadtheDocs and Wiki
- Versioning – need standards established

ONAP API Documentation – Beijing Goals

1. Consistent format & structure across projects

- Use **OpenAPI spec** (formerly known as **Swagger**)
- Standard set of attributes & definitions (e.g. Name, location, Type, Description)
 - Clear and concise summary of each API, Primary object(s), and list of Consumers
 - Differentiate RESTful from Healthcheck and Component APIs

#1 Format - OpenAPI Specification

OpenAPI Specification –OAI/S (aka swagger spec) <https://www.openapis.org/about>

- Helps define the API's contract for API development teams and its end consumers
- Sets clear expectations for the experience of integrating with the API
- Provides language agnostic human-readable format for describing API operations

Background

The Open API Initiative (OAI) is as an open governance structure under the Linux Foundation. It was created by a consortium of forward-looking industry experts who recognized the value of standardizing on how REST APIs are described. The OAI is focused on creating, evolving and promoting a vendor neutral description format for APIs, regardless of the language they are developed and implemented in. SmartBear Software donated the Swagger Specification directly to the OAI as the basis of this Open Specification.

ONAP API Documentation – Beijing Goals

1. Consistent format & structure across projects

- Use **OpenAPI spec** (formerly known as **Swagger**)
- Standard set of attributes & definitions (e.g. Name, location, Type, Description)
 - Clear and concise summary of each API, Primary object(s), and list of Consumers
 - Differentiate RESTful from Healthcheck and Component APIs

2. ReadTheDocs as central repository for all API documentation

- Examples: Swagger.json, Postman collections, HTTP Headers (params/values), etc.
- Additional functional descriptions and interfaces as needed

#2 ReadtheDocs – Current API documentation

ONAP Component	API name (offered API only)	pdf	Rst/html	Swagger.json on readthedocs	External Swagger UI	Postman Collection	Swagger.json and swagger UI on ONAP instance
CLAMP	CLAMP API	No	only an healthcheck	No	No	No	No
SDC	External API	No	Presentation to be improved, basic descriptions	No	No	No	http://sdc.api.simpledemo.onap.org:8080/api-docs/
APPC	APPC LCM API	No	lot of non API-related doc	No	No	No	http://appc.api.simpledemo.onap.org:8282/apidoc/explorer/index.html#!/appc-provider-lcm(2016-01-08)
APPC	APPC OAM API	No	lot of non API-related doc	No	No	No	http://appc.api.simpledemo.onap.org:8282/apidoc/explorer/index.html#!/appc-oam(2017-03-03)
DCAE	CDAP Broker API	No	basic, Presentation to be improved	No	No	No	?
DCAE	Config Binding Service	No	basic, Presentation to be improved	No	No	No	?

#2 ReadtheDocs – Current API documentation, cont'd

ONAP Component	API name (offered API only)	pdf	Rst/html Quality (poor/medium/good)	Swagger.json on readthedocs	External Swagger UI	Postman Collection	Swagger.json and swagger UI on ONAP instance
DCAE	Deployment-handler API	No	basic, Presentation to be improved	No	No	No	?
DCAE	DCAE Inventory API	No	basic, Presentation to be improved	No	No	No	?
DCAE	VES collector	No	basic, Presentation to be improved	No	No	No	?
HOLMES	Holmes API	No	basic, Presentation to be improved	No	No	No	?
POLICY	Policy Engine service	No	contents menu to be improved,	No	No	No	http://10.4.2.24:8081/pdp/swagger-ui.html#/
POLICY	Policy API details	No	lot of things are missing (configPolicyAPIRequest ?, deletePolicyParameters ?...)	No	No	No	?

#2 ReadtheDocs – Current API documentation, cont'd

ONAP Component	API name	pdf	Rst/html Quality (poor/medium/good)	Swagger.json on readthedocs	External Swagger UI	Postman Collection	Swagger.json and swagger UI on ONAP instance
SO	Northbound API	No	no explanation, lot of southbound things (no real interest)	No	No	No	?
VFC	NSLCM API	No	presentation to be improved	No	No	No	?
VFC	VNFM Driver API	No	southbound API	No	No	No	?
VFC	VNF LCM API	No	northbound ??	No	No	No	?
AAI	AAI REST API	No	nothing about operations nor resources...	No	No	No	?
DMaaP	Message Router API ?	No	presentation to be improved	No	No	No	?
Modeling Parsers	Checker API	No	lack of documentation	No	No	No	?
Modeling Parsers	NFV Parser API	No	Nothing	No	No	No	?
MSB	MSB API ?	No	Nothing	No	No	No	?
MultiVim	MultiCloud API	No	presentation to be improved	No	No	No	?
VNF SDK	Market Place API	No	lack of description	No	No	No	?

#2 ReadtheDocs – Current API documentation, cont'd

ONAP Component	API name	Link on wiki	Swagger on ONP instance
DCAE	DCAE controller	https://wiki.onap.org/display/DW/DCAE+API	
DCAE	DCAE Message router	https://wiki.onap.org/display/DW/DCAE+API	
Portal	Portal API	https://wiki.onap.org/display/DW/Portal+API	
SDC	Internal API		http://sdc.api.simpledemo.onap.org:8080/api-docs/
?	AAF API	https://wiki.onap.org/display/DW/AAF+API	
DMaaP	Data Router API	https://wiki.onap.org/display/DW/DMaaP+API	
?	Modeling API : 3 parsers	https://wiki.onap.org/display/DW/Modeling+API	
SO	ARIA Rest API	https://wiki.onap.org/display/DW/ARIA+REST+API	
AAI	External System Operation	https://wiki.onap.org/pages/viewpage.action?pagelId=11930343	
AAI	External System Register	https://wiki.onap.org/pages/viewpage.action?pagelId=15999660	
SDNC	SLI-API, GENERIC-RESOURCE-API, VNF-API		http://10.4.2.11:8282/apidoc/explorer/index.html

#2 ReadtheDocs – Element examples

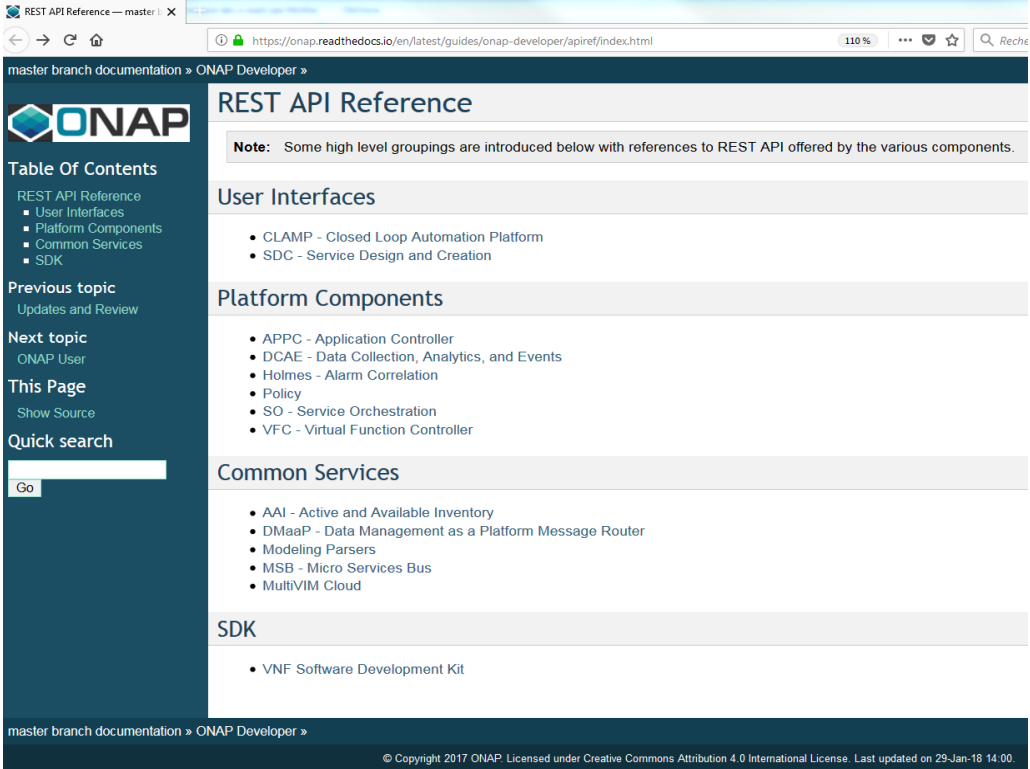
Swagger.json Options:

- Swagger json file displayed by Sphinx directive ..
Swaggerv2doc (*Note: If projects need to add information, an equivalent / ReStructured text version can be created*)
- Actual json file referenced in a nexus.onap.org artifact repo (probably raw site)
- Reference to a specific gerrit repo branch/verision

Postman Collections

- Export from postman tools, commit to gerrit, release in a nexus.onap.org raw site and then reference from readthedocs
- Not required for all projects

HTTP Headers



The screenshot shows a web browser displaying the ONAP REST API Reference documentation. The page title is "REST API Reference" and the URL is "https://onap.readthedocs.io/en/latest/guides/onap-developer/apiref/index.html". The page content is organized into several sections:

- Table Of Contents:** REST API Reference, User Interfaces, Platform Components, Common Services, SDK.
- Previous topic:** Updates and Review.
- Next topic:** ONAP User.
- This Page:** Show Source.
- Quick search:** A search bar with a "Go" button.
- REST API Reference:** A note stating: "Some high level groupings are introduced below with references to REST API offered by the various components."
- User Interfaces:** A list of components: CLAMP - Closed Loop Automation Platform, SDC - Service Design and Creation.
- Platform Components:** A list of components: APPC - Application Controller, DCAE - Data Collection, Analytics, and Events, Holmes - Alarm Correlation, Policy, SO - Service Orchestration, VFC - Virtual Function Controller.
- Common Services:** A list of services: AAI - Active and Available Inventory, DMaaP - Data Management as a Platform Message Router, Modeling Parsers, MSB - Micro Services Bus, MultiVIM Cloud.
- SDK:** A list of SDKs: VNF Software Development Kit.

The footer of the page contains the text: "© Copyright 2017 ONAP. Licensed under Creative Commons Attribution 4.0 International License. Last updated on 29-Jan-18 14:00."

ONAP API Documentation – Beijing Goals

1. Consistent format & structure across projects

- Use **OpenAPI spec** (formerly known as **Swagger**)
- Standard set of attributes & definitions (e.g. Name, location, Type, Description)
 - Clear and concise summary of each API, Primary object(s), and list of Consumers
 - Differentiate RESTful from Healthcheck and Component APIs

2. ReadTheDocs as central repository for all API documentation

- Examples: Swagger.json, Postman collections, HTTP Headers (params/values), etc.
- Additional functional descriptions and interfaces as needed

3. Versioning

- Need more prescriptive naming standards & version control
- Possible recommendations include: Semantics (e.g. “Major, Minor, Patch”), Backwards Compatibility (BWC) Policy, and Custom Headers

ONAP API Versioning

Potential recommendations* for standardizing OAP APIs:

- **Standard semantics** (e.g. “Major, Minor, Patch”) would ensure all APIs will be “speaking the same language”
- **Backwards compatibility policy** would define / limit how long previous versions need to be active/available
- **URL Structure** and **Custom headers** would allow API clients to target specific versions and servers and evolve APIs without breaking existing clients

** Subject to PTL support and TSC approval*

ONAP API Documentation – Other PTL / TSC Questions

1. Use of **SwaggerHub** for E2E API Design, Build, Documentation, and Deployment?

<u>Benefits</u>	<u>Costs</u>
✓ Cloud-Based Hosting	○ Funding
✓ Auto synching with GitHub, etc.	○ Set-up & Training
✓ Collaboration	○ Changing current workflows
✓ Workflow	
✓ Versioning	

2. Hosting options

- SwaggerHub
- CloudCode
- Other?

3. Model enabled vs. API driven approaches

- Are both appropriate?
- When should we use one versus the other?

ONAP API Documentation – SwaggerHub Description

Swaggerhub components:

- **Swagger UI** – human readable version provides interaction with API without coding
- **Swagger Editor** – document and validate API according to OAS
- **Swaggerhub** – team collaboration, maintains secure version repository, integration to other dev and management tools
- **Swagger Codegen** – generate client/server sample code in 30+ programming languages

SwaggerHub is an integrated API Development platform, built for teams, that brings the core capabilities of the Swagger framework to design, build, document and deploy APIs. SwaggerHub enables development teams to collaborate and coordinate the entire lifecycle of an API with the flexibility to integrate with the toolset of your choice.

- <http://app.swaggerhub.com>

ONAP API Documentation – SwaggerHub Features



Design	?	Cloud-based YAML editor with predesigned templates, simultaneous style feedback, mocking, and hosting for API definitions
Develop	?	Code generation in 20+ server languages, which can automatically sync with source control hosts like GitHub, Bitbucket, and GitLab
Document	★	Interactive API documentation and sandbox, and auto-generation for client SDKs, all hosted in the cloud with privacy controls
Mock	?	Generate a mock of your API to virtualize operations without writing any code
Host	★	Secure cloud-based hosting for Swagger definitions and API documentation, with built-in access controls
Collaborate	?	Real-time issue tracking and conversations, team management, role assignment, and permission/access controls
Deploy	★	Quickly deploy your API definitions to API Gateways such as AWS , Microsoft Azure and IBM API Connect. Automatically generate AWS Lambda functions for serverless deployments.
Version	★	Manage multiple versions of API definitions, and retire outdated APIs

ONAP API Documentation – SwaggerHub Benefits

- **Cloud-Based Editor with Advanced Design Capabilities**
 - Write and edit your Swagger definition in a cloud-based editor with built-in style validation to reduce errors, and auto-mocking that lets you preview how your API will behave and validate design decisions without writing any code.
- **Integrated Documentation Workflow**
 - Write your Swagger definition, work with your documentation, and generate code and client SDKs in one intuitive interface, without the need to work across multiple tools. SwaggerHub keeps everything in sync and seamlessly integrates with source control and API management platforms.
- **Secure Cloud-Hosting**
 - Store all of your API definitions, and associated documentation, in a platform built for APIs. SwaggerHub auto-saves your work throughout the design process and provides a central place to host your documentation, without the need to setup a server.
- **Privacy and Access Control**
 - Control who can access your APIs, and keep sensitive information secure with SwaggerHub's built-in privacy controls. Use access controls when designing new APIs, or documenting an existing Swagger definition.
- **Versioning and Publishing**
 - Update and iterate continuously, with the ability to manage multiple versions of your API in SwaggerHub. New versions copy existing syntax from the last updated version so you'll never have to start building from scratch.
- **Collaboration & Team Management**
 - SwaggerHub provides a central platform for teams to collaborate throughout the design and documentation of your API. Securely share your APIs with internal and external contributors, and track issues in real-time with comments in the SwaggerHub editor. SwaggerHub will automatically notify collaborators whenever a change is made the definition.
- **Sync with Source Control and API Management Platforms**
 - Manage the entire lifecycle of your API from a central platform, which syncs the different versions of your API definitions with source control tools and API management platforms — allowing you to update your Swagger definition and automatically update your documentation, server code, and deployment.
- **Reuse Common Models**
 - Store all your common components in Domains, which can then be referenced from all your APIs, standardizing work and saving time and effort.

ONAP API Documentation – SwaggerHub Pre-Planning

1. Identify team –API clients, API providers, API governance

- Sign up for team or enterprise plan
https://app.swaggerhub.com/prices?_ga=2.71334490.893424181.1517237482-936456995.1511884392

2. Establish process

- **PayPal example:**

3. Issue guidelines

- Security
- Naming of resource, parameters, responses
- Error handling
- Data format handling
- Versioning
- **Zalando example:** <http://zalando.github.io/restful-api-guidelines/index.html#general-guidelines>

ONAP API Documentation – SwaggerHub Set-up

- **Setup environment**
 - Define Organizations and Teams
 - Create APIs under Organizations
 - Invite Collaborators
- **Training**
 - Create, import, document, version, fork, merge API
 - Sync API definitions with GitHub, GitLab, Bitbucket
 - Mock API
 - Generate server stub and client SDK

Details for above steps: <https://app.swaggerhub.com/help/tutorials/getting-started>

ONAP API Documentation – SwaggerHub API-1st approach

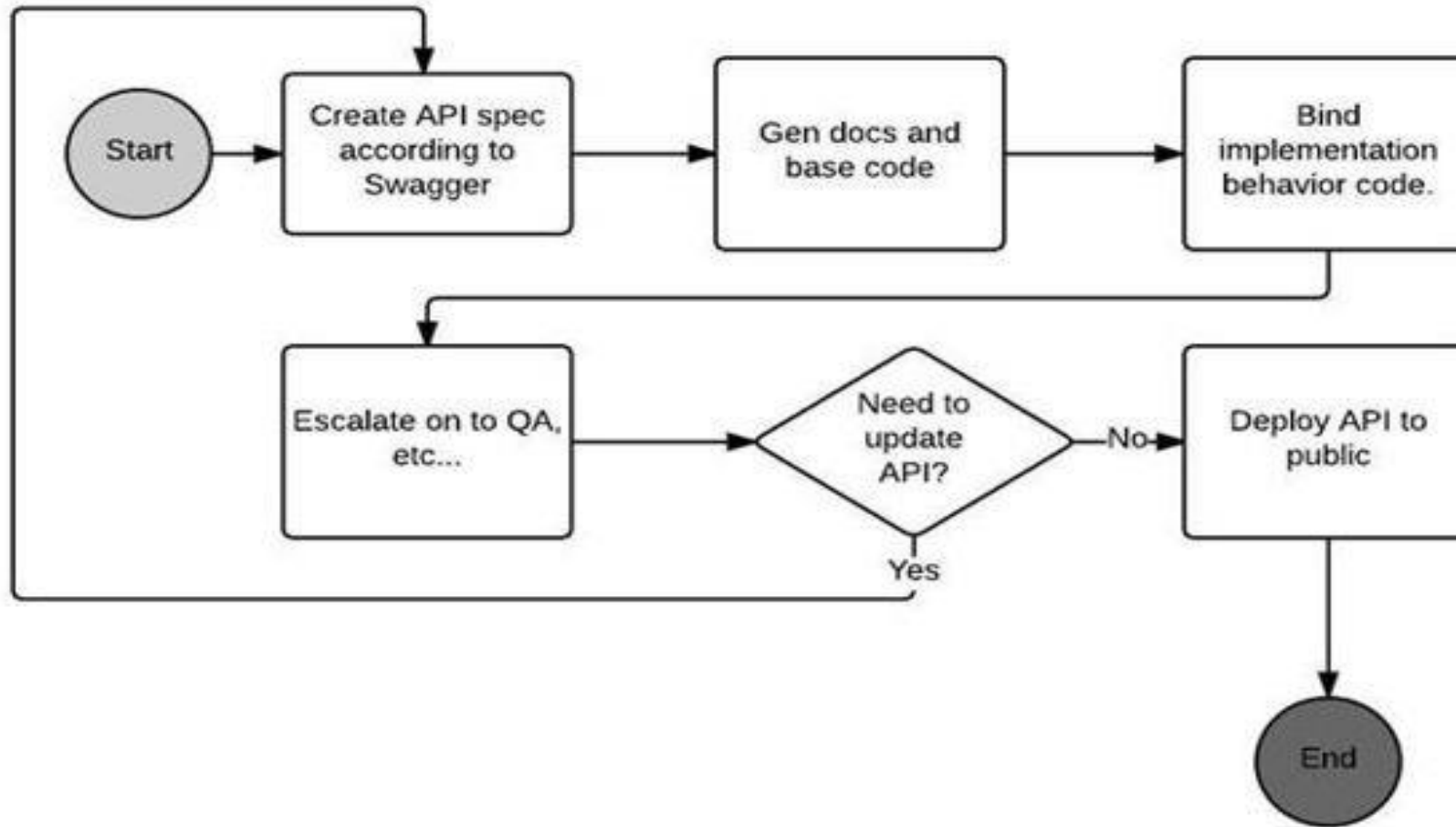
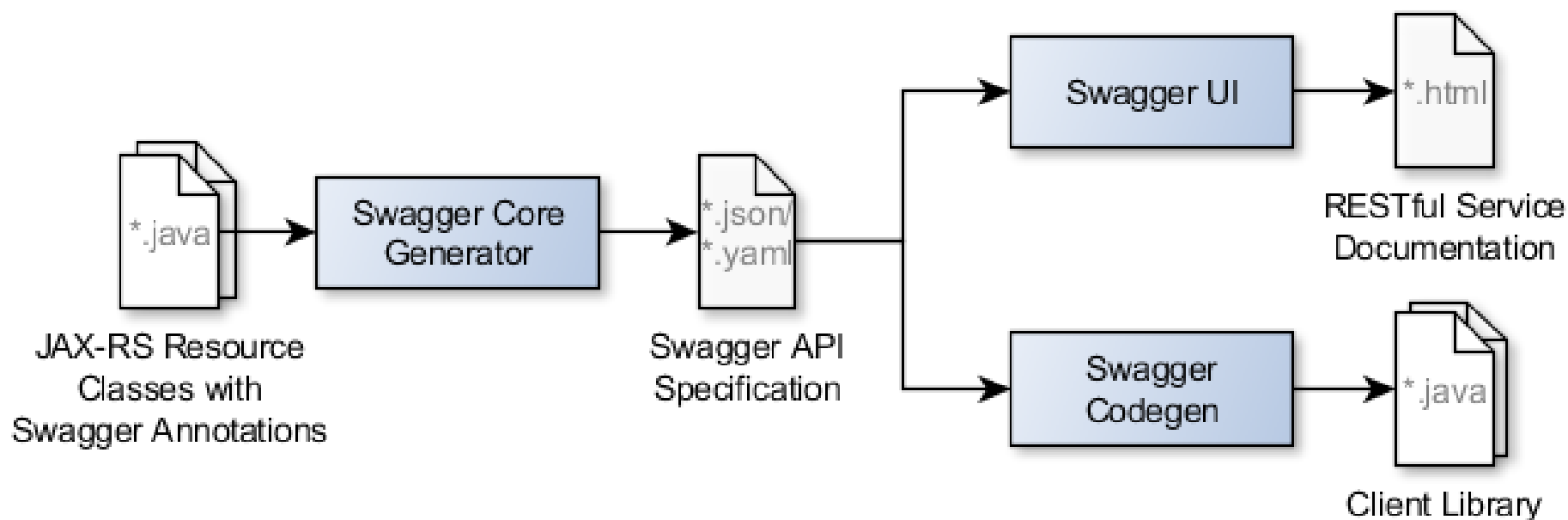


Diagram from: <https://www.codeguru.com/cpp/frameworks/the-value-of-specification-first-development-using-swagger.html>

ONAP API Documentation – SwaggerHub Code-1st approach



<https://blog.philippbauer.de/enriching-restful-services-swagger/>