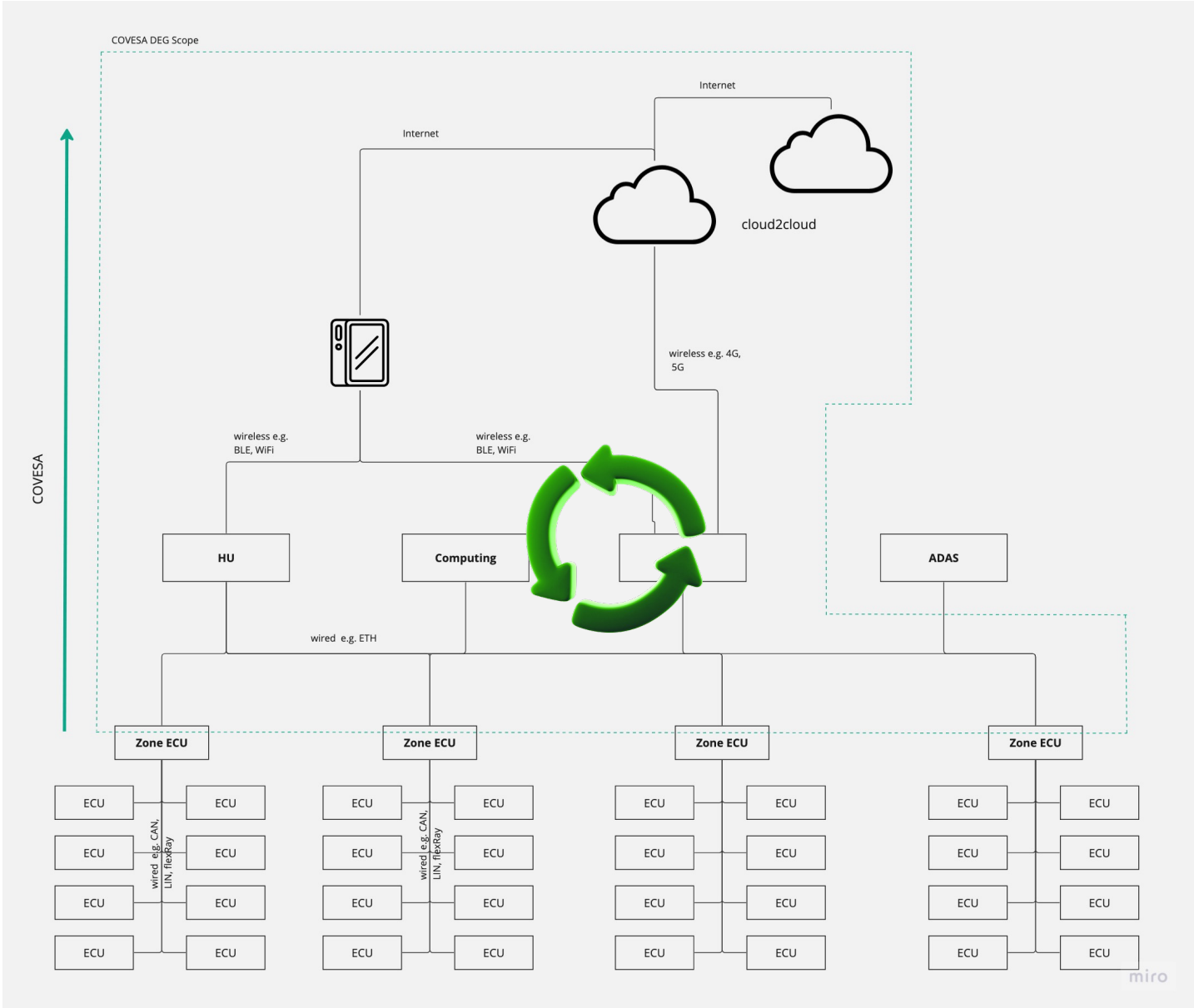
OpenAPI/AsyncAPI and VSS

COVESA

Accelerating the future of connected vehicles


COVESA DEG Scope
Internet
Internet
cloud2cloud
wireless e.g. 4G, 5G
wireless e.g. BLE, WiFi
wireless e.g. BLE, WiFi
COVESA
HU
Computing
ADAS
wired e.g. ETH
Zone ECU
Zone ECU
Zone ECU
Zone ECU
ECU
wired e.g. CAN, LIN, flexRay
miro

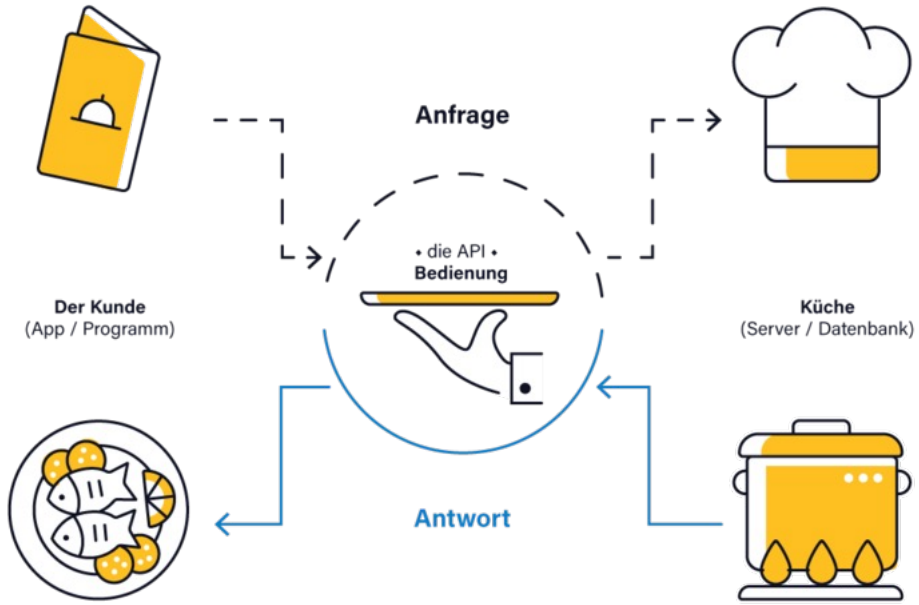
SCOPE:

- Domain specific interfaces.
- Cross-Domaine interfaces.
- Vehicle function interfaces.
- Avoid legacy interfaces/specs!

WHY do we need IDLs

IDL is a language-independent specification language used to define the interfaces and contracts of components in a distributed system

- It defines the structure and **behavior of services, including the data types, methods, and communication protocols** used for interaction.
- It focuses on specifying the **structure and semantics of the components'** interfaces **without specifying the underlying communication** mechanisms.
- It **provides a common ground for different programming languages** to understand and interact with each other by defining a shared interface.
- It can be used **can be used to generate language-specific code**, such as client and server stubs, which provide the necessary plumbing code.



COVESA

PATTERN

Client-Server - This is the most common communication architecture, where clients request services or resources from a centralized server.

Service Oriented (SOA) - SOA is an architectural style that focuses on loosely coupled services that communicate with each other over a network.

Message-oriented (MOM) - MOM is an architecture that uses asynchronous message passing between components.

Event-Driven (EDA) - consists of event producers that generate a stream of events, and event consumers that listen for the events

….. and many others.

Overview of industry standards.

Protobuf, simple language for serializing data and defining interfaces. Designed by google team, ideal for client-server pattern.

Thrift, language and binary communication protocol used for defining and creating services, designed by Facebook, ideal for client-server pattern.

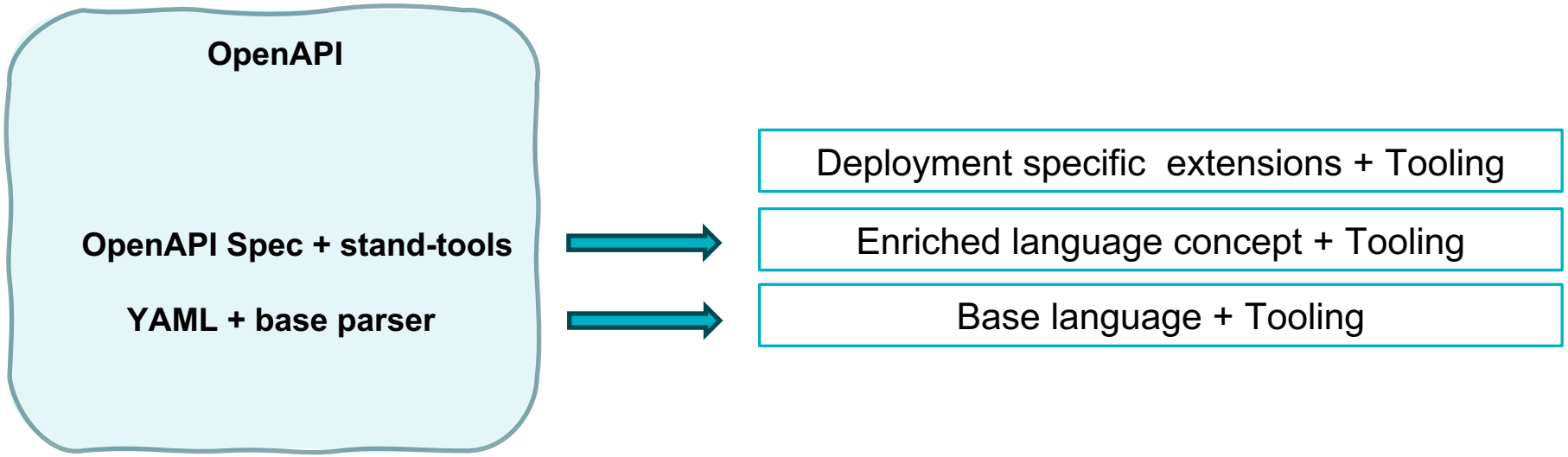
OpenAPI (ex. Swagger)- is a specification for a machine-readable interface definition language for describing, producing, consuming and visualizing services. Designed by OPENApi initiative, ideal for client-server pattern. 26k stars.

API Blueprint, is built to encourage dialogue and collaboration between project stakeholders, developers and customers at any point in the API lifecycle. 8.5k stars.

AsyncAPI - The AsyncAPI specification allows you to create machine-readable definitions of your asynchronous APIs. Designed by asyncAPI intiative, 3.1k stars. Ideal for events, messages. Coming from OpenAPI.

and we go on ……

How this is layered!



OPEN API.

Sync calls in one-to-one relation, and it supports overlay spec.

Numbers:

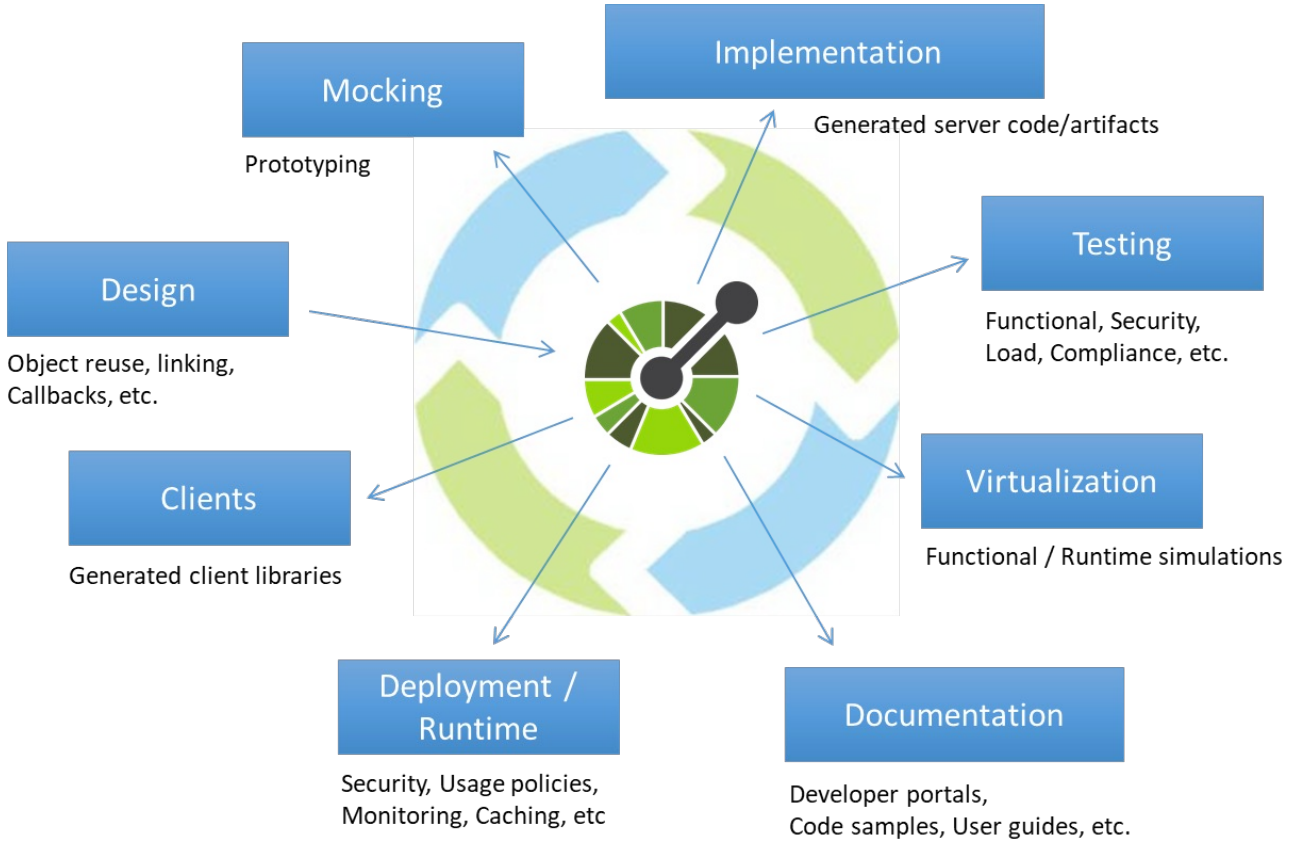
- 26.6k stars
- 9.1k forks

Tools:

Auto Generators, Converters, Data Validators, Description Validators, Documentations, DSL, Gateways, GUI Editors, Mock Servers, Parsers, SDK Generators, Testing, Plugins for IDEs

Programming Languages:

10+



ASYNC API.

Async calls in one-to-one and one-to-many relation.

GOAL is to support sync calls as well.

Numbers:

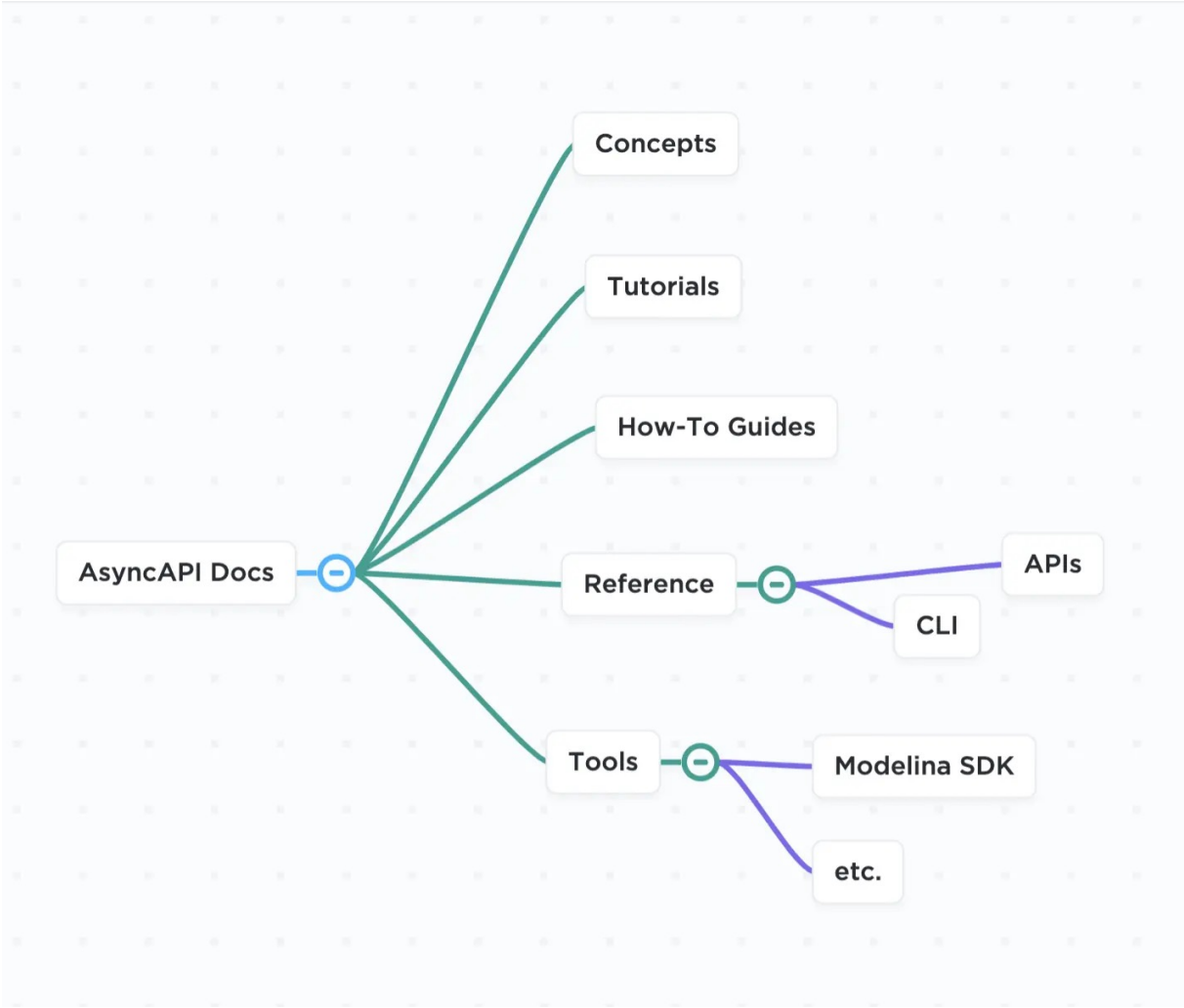
- 3.1k stars
- 228 forks

Tools:

Converters, Bindings, Data Validators, Description Validators, Data Modelling, Documentations, GUI Editors, Mock Servers, Parsers, SDK Generators, Testing, Open API Parser, Plugins for IDEs, Simulator

Programming Languages:

7+



OpenAPI < ==== > AsyncAPI

ASYNCAPI vs OPENAPI

OpenAPI is focused on designing and documenting HTTP APIs, while AsyncAPI is designed for asynchronous APIs and event-driven architectures.

Files are compatible and goal is to bring two standards together.

EXAMPLE with ASYNC API.

```
17          resourcePath:
18            default: vehicle/cabin/hvac
19          version:
20            default: v1
21    channels:
22      /methodStartHVAC:
23        publish:
24          summary: Channel for updating the HVAC settings
25          operationId: setHVACrequest
26          message:
27            oneOf:
28              - $ref: '#/components/messages/vehicle-cabin-HVAC-defrostLevel'
29              - $ref: '#/components/messages/vehicle-cabin-HVAC-circulationLevel'
30              - $ref: '#/components/messages/vehicle-cabin-HVAC-FrontDefroster-isActive'
31              - $ref: '#/components/messages/vehicle-cabin-HVAC-isACtive'
32        subscribe:
```

Each element is modeled with right namespace, range and datatype that can be used for validation purposes.

Same description can be used one-to-one for data architecture needs. Data storage, modelling, streaming etc.

EXAMPLE with OPEN API.

```yaml
paths:
  /vehicle/acceleration:
    post:
      summary: Provide Vehicle Acceleration Data
      description: Endpoint to send vehicle acceleration data.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: schema/mydemo_vehicle.schema.json#/Vehicle/$children/Acceleration
      responses:
        '200':
          description: Success. Acceleration data received.
        '400':
          description: Bad Request. Invalid payload or data format.

  /vehicle/acceleration/mySpecificLocationEvent:
    get:
      summary: Get Current Vehicle Data
      description: Get the current location and acceleration data of the vehicle.
      responses:
        '200':
          description: Success. Current vehicle data received.
          content:
            application/json:
              schema:
                type: object
                properties:
                  Location:
                    $ref: schema/mydemo_vehicle.schema.yaml#/Vehicle/$children/CurrentLocation
                  Acceleration:
                    $ref: schema/mydemo_vehicle.schema.yaml#/Vehicle/$children/Acceleration
```

Check PR https://github.com/COVESA/vss-tools/pull/296

COVESA

Accelerating the future of connected vehicles