asyncapi.com/docs

ASYNCAPI

Proud to be part of the Linux Foundation

asyncapi.com/cheatsheet

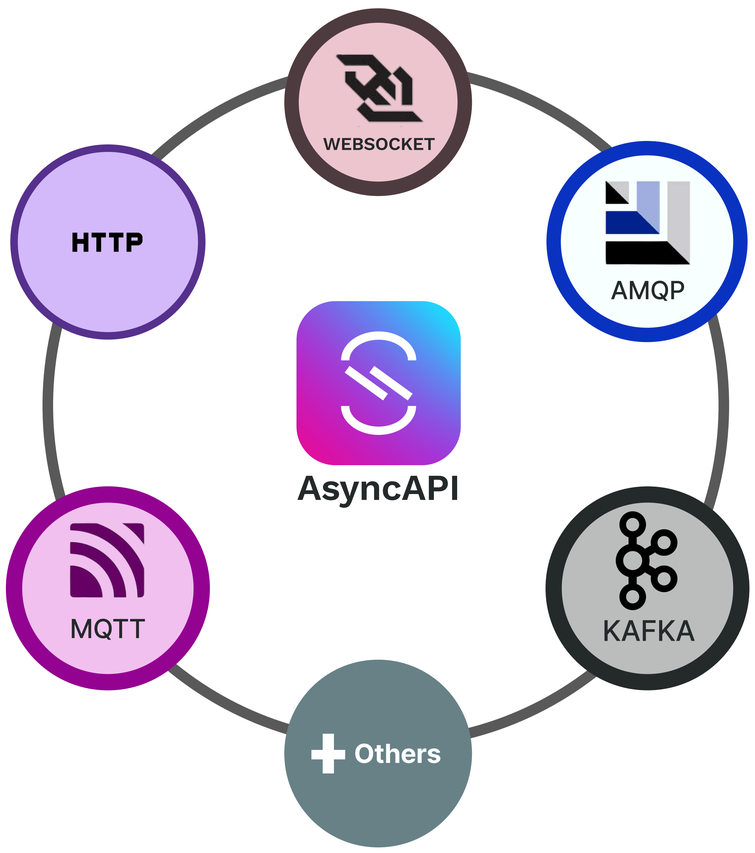
Open Source Specification

- It is an **open source specification** that makes it easier to work with even-driven architectures by helping with design, documentation and managing of asynchronous APIs,

Can be described with **AsyncAPI**

Protocol Agnostic

- It is protocol agnostic, and you can use it with protocols like **MQTT, Kafka, AMQP, and WebSocket, just to name a few.**



Machine-Readable Format

- It can be represented with machine-readable formats like **YAML** and **JSON.**

```
asyncapi : ...

info :
  title :  ...
  version : ...

server :
  prod :
    host :  ...
    protocol :  ...
   ...

channels :
  myTopic :
    address :  ...
    messages :  { ... }
   ...

operations :
  sendMsg :
      channel :  ...
      action :  ...
    ...

components :
    schemas :  { ... }
   ...
```

Tools

AsyncAPI is more than just a specification; it's a vibrant community of enthusiasts dedicated to developing a suite of open-source tools that support the specification and facilitate its seamless adoption.

asyncapi.com/tools

Protocol Bindings

- Additional protocol-specific information you can add to AsyncAPI document

Binding - Channel

```yaml
# ... (redacted for brevity)
channels :
  userSignedUp :
    address : user_signedup
    messages :
      userSignedUp :
        $ref : '#/components/messages/userSignedUp'
    bindings :
      kafka :
        bindingVersion : '0.5.0'
        partitions : 10
        replicas : 2
        topicConfiguration :
          cleanup.policy : ["delete", "compact"]
```

Binding - Operation

```yaml
# ... (redacted for brevity)
operations :
  userSignup :
    action : receive
    channel :
      $ref : '#/channels/userSignedUp'
    bindings :
      mqtt :
        qos : 2
        retain : true
        bindingVersion : 0.2.0
```

Binding - Message

```yaml
# ... (redacted for brevity)
components :
  messages :
    userSignedUp :
      payload :
        $ref : '#/components/schemas/userSignedUp'
      bindings :
        amqp :
          contentEncoding : gzip
          messageType : 'user.signup'
          bindingVersion : 0.3.0
```

Binding - Server

```yaml
# ... (redacted for brevity)
servers :
  production:
    bindings:
      pulsar:
        tenant: contoso
        bindingVersion: '0.1.0'
```

Multi Format Schema

- By default you can use AsyncAPI Schema that is a superset of JSON Schema Draft 7. Multi format allows you to use other formats, like **OpenAPI Schema, Apache Avro, Protobuf and others.**

Default - AsyncAPI Schema

```yaml
# ... (redacted for brevity)
components :
  schemas :
    userSignedUp :
      type : object
      properties :
        userId :
          type : integer
          description : This property describes the id of the user
        userEmail :
          type : string
          description : This property describes the email of the user
```

Avro Example

```yaml
# ... (redacted for brevity)
components :
  schemas :
    userSignedUp :
      schemaFormat : 'application/vnd.apache.avro;version=1.9.0'
      schema:
       type: record
       name: UserSignedUp
       namespace: com.company
       doc: User sign-up information
       fields:
         - name: userId
           type: int
         - name: userEmail
           type: string
```

Protobuf Example

```yaml
# ... (redacted for brevity)
components :
  schemas :
    userSignedUp :
      schemaFormat: 'application/vnd.google.protobuf;version=3'
      schema : |
        message UserSignedUp {
          required int32 user_id = 1;
          optional string user_email = 2;
        }
```

Extensibility

- With **x- prefix** you can add additional fields to AsyncAPI documents to fulfil your specific use cases

```yaml
asyncapi: 3.0.0
info:
  title: Cool Example
  version: 0.1.0
  x-linkedin: 'https://www.linkedin.com/company/asyncapi'
```

Reusability

- You can reuse all parts of AsyncAPI, within one document or by pointing to external resources, separate files or servers that for example can host schemas.

URL

```yaml
# ... (redacted for brevity)
components :
  messages :
    UserSignedUp :
      payload :
        $ref:  '#/components/schemas/UserSignedUp'
  schemas :
    UserSignedUp :
      $ref:  http://localhost:8080/apis/registry/v2/groups/my-group/artifacts/UserSignedUp
```

Single document

```yaml
# ... (redacted for brevity)
components :
  messages :
    UserSignedUp :
      payload :
        $ref : '#/components/schemas/UserSignedUp'
  schemas :
    UserSignedUp :
      type : object
      properties :
        displayName :
          type : string
          description : Name of the user
        email :
          type : string
          format : email
          description : Email of the user
```

To a file

```yaml
# ... (redacted for brevity)
components :
  messages :
    UserSignedUp :
      payload :
        $ref: '#/components/schemas/UserSignedUp'
  schemas :
    UserSignedUp :
      $ref : './userSchema.json'
```