Publishing to AsyncAPI

You can convert events in CDS models to the AsyncAPI specification , a widely adopted standard used to describe and document message-driven asynchronous APIs.

Table of Contents

- Usage from CLI
- Presets
- Annotations
- Extensions
  - Behavior with --merged flag
- Type Mapping

Usage from CLI

Use the following command to convert all services in `srv/` and store the generated AsyncAPI documents in the `docs/` folder:

```sh
cds compile srv --service all -o docs --to asyncapi
```

For each service that is available in the `srv/` files, an AsyncAPI document with the service name is generated in the output folder. If you want to generate one AsyncAPI document for all the services, you can use `--asyncapi:merged` flag:

```sh
cds compile srv --service all -o docs --to asyncapi --asyncapi:merged
```

↳ *Learn how to programmatically convert the CSN file into an AsyncAPI Document*

Presets

Use presets to add configuration for the AsyncAPI export tooling.

.cdsrc.json

```json
{
  "export": {
    "asyncapi": {
      "application_namespace": "sap.example"
      [...]
    }
  }
}
```

Term	Preset Target	AsyncAPI field	Remarks
merged.title	Service	info.title	Mandatory when *--asyncapi:merged* flag is given. *title* from here is used in the generated AsyncAPI document.
merged.version	Service	info.version	Mandatory when *--asyncapi:merged* flag is given. *version* from here is used in the generated AsyncAPI document
merged.description	Service	info.description	Optional when *--asyncapi:merged* flag is given. *description* from here is

Term	Preset Target	AsyncAPI field	Remarks
			used in the generated AsyncAPI document.
merged.short_text	Service	x-sap-shortText	Optional when `--asyncapi:merged` flag is given. The value from here is used in the generated AsyncAPI document.
application_namespace	Document	x-sap-application-namespace	Mandatory
event_spec_version	Event	x-sap-event-spec-version	
event_source	Event	x-sap-event-source	
event_source_params	Event	x-sap-event-source-parameters	
event_characteristics	Event	x-sap-event-characteristics	

Annotations

Use annotations to add configuration for the AsyncAPI export tooling.

> **TIP**
>
> Annotations will take precedence over presets.

Term (*@AsyncAPI.*)	Annotation Target	AsyncAPI field	Remarks
Title	Service	info.title	Mandatory

Term (*@AsyncAPI.*)	Annotation Target	AsyncAPI field	Remarks
SchemaVersion	Service	info.version	Mandatory
Description	Service	info.description	
StateInfo	Service	x-sap-stateInfo	
ShortText	Service	x-sap-shortText	
EventSpecVersion	Event	x-sap-event-spec-version	
EventSource	Event	x-sap-event-source	
EventSourceParams	Event	x-sap-event-source-parameters	
EventCharacteristics	Event	x-sap-event-characteristics	
EventStateInfo	Event	x-sap-stateInfo	
EventSchemaVersion	Event	x-sap-event-version	
EventType	Event		Optional; The value from this annotation will be used to overwrite the default event type in the AsyncAPI document.

For example:

```cds
@AsyncAPI.Title        : 'CatalogService Events'
@AsyncAPI.SchemaVersion: '1.0.0'
@AsyncAPI.Description  : 'Events emitted by the CatalogService.'

service CatalogService {
  @AsyncAPI.EventSpecVersion    : '2.0'
  @AsyncAPI.EventCharacteristics: {
    ![state-transfer]: 'full-after-image'
  }
  @AsyncAPI.EventSchemaVersion       : '1.0.0'
```

```
  event SampleEntity.Changed.v1 : projection on CatalogService.SampleEntity;
}
```

Extensions

`@AsyncAPI.Extensions` can be used to provide arbitrary extensions. If a specific annotation exists for a given extension, it takes precedence over the definition using @AsyncAPI.Extensions. For example, if both `@AsyncAPI.ShortText` and `@AsyncAPI.Extensions: { ![sap-shortText]: 'baz' }` are provided, the value from `@AsyncAPI.ShortText` will override the one defined in @AsyncAPI.Extensions.

For example:

```cds
@AsyncAPI.Extensions   : {
  ![foo-bar]                   : 'baz',
  ![sap-shortText]             : 'Service Base 1'
}

service CatalogService {
  @AsyncAPI.Extensions         : {
    ![sap-event-source]          : '/{region}/sap.app.test'
  }
  event SampleEntity.Changed.v1 : projection on CatalogService.SampleEntity;
}
```

The `@AsyncAPI.Extensions` annotation can be applied at both the service level and the event level.

Since the AsyncAPI specification requires all extensions to be prefixed with `x-`, the compiler will automatically add this prefix. Therefore, do not include the `x-` prefix when specifying extensions in `@AsyncAPI.Extensions`.

Behavior with `--merged` flag

When the `--merged` CLI flag is used:

- Extensions defined via `@AsyncAPI.Extensions` on `services` are **ignored**.
- Extensions defined via `@AsyncAPI.Extensions` on `events` remain effective and are applied as expected.

Type Mapping

CDS Type to AsyncAPI Mapping

CDS Type	AsyncAPI Supported Types
`UUID`	`{ "type": "string", "format": "uuid" }`
`Boolean`	`{ "type": "boolean" }`
`Integer`	`{ "type": "integer" }`
`Integer64`	`{ "type": "string", "format": "int64" }`
`Decimal , {precision, scale}`	`{ "type": "string", "format": "decimal", "x-sap-precision": <precision>, "x-sap-scale": <scale> }`
`Decimal` , without scale	`{ "type": "string", "format": "decimal", "x-sap-precision": <precision> }`
`Decimal` , without precision and scale	`{ "type": "string", "format": "decimal" }`
`Double`	`{ "type": "number" }`
`Date`	`{ "type": "string", "format": "date" }`
`Time`	`{ "type": "string", "format": "partial-time" }`
`DateTime`	`{ "type": "string", "format": "date-time" }`
`Timestamp`	`{ "type": "string", "format": "date-time" }`
`String , {maxLength}`	`{ "type": "string", "maxLength": length }`
`Binary , {maxLength}`	`{ "type": "string", "maxLength": length }`
`LargeBinary`	`{ "type": "string" }`

CDS Type	AsyncAPI Supported Types
`LargeString`	`{ "type": "string" }`

Edit this page

Last updated: 13/08/2025, 06:44

Previous page
OpenAPI

Next page
Serving UIs