Guide to Implementing Criterion OpenAPI Specification

Version: **1.1**
Date: **1 5 2024**
Distribution: **Industry**
Document Name: **Guide_to_Implementing_Criterion_OpenAPI_Specification.pdf**

DISCLAIMER

Criterion believes it has employed personnel using reasonable skill and care in the creation of this document. However, this document is provided to the reader 'as is' without any warranty (express or implied) as to accuracy or completeness. Criterion cannot be held liable for any errors or omissions in this document or any losses, damages or expenses arising consequent to the use of this document by the reader.

CHANGE HISTORY

DATE	VERSION	DESCRIPTION
29 November 2023	1.0	Initial guide.
1 May 2024	1.1	Following consultation with STAG, update approaches to end point versioning, security and extensibility.

CONTENTS

1 HOW TO USE CRITERION OPENAPI SPECIFICATION.

Criterion do not host services. The host of these services will be defined as 'service provider' and the parties calling the end point will be defined as 'service client'. Middle third parties are seen as being both service provider and service client. The added value of these third parties is generally out of scope for Criterion. However, Criterion does recognise the value there.

1.1 SERVICE PROVIDER

The OpenAPI Specification is designed to be downloaded, edited and then hosted by the service provider.

The service provider must:

- Keep reference to Criterion, including links and copyright information;
- The Criterion version of the resources must be included alongside each endpoint.

The service provider should:

- Use the version number in the server section to reflect the service provider's internal versioning;
- Keep the Criterion tag names;
- Not change any of the data structures or their meaning (see section *TRADING PARTNER SPECIFIC DATA* below);
- Send a copy of the adapted OAS (OpenAPI Specification) to Criterion to help aid future development (under NDA if required);
- Inform Criterion of any problems found;
- Make their versions of adapted standards available to their clients (under limitations set out in the licensing contract);
- Use the hypermedia links where provided as a primary way to extend and layout the offering;
- Where REST IDs are defined, these should be used as surrogates to human identifiable identifiers.

The service provider could:

- Add in digital signatures;
- Add additional commentary to descriptions;
- Set up an OAS SwaggerUI (or other) test interface for use by their clients (under limitations set out in the licensing contract);
- Provide examples of usage (under limitations set out in the licensing contract);
- Integrate the Criterion OAS with other OAS in the same domain when appropriate.

1.2 SERVICE CLIENT

The client should:

- Dominantly use the OAS of the service providers not Criterion's OAS directly in implementation;
- Use Criterion's OAS as a Rosetta stone when dealing with multiple service providers;
- Use Criterion's OAS when initially evaluating the need for the Standard;
- Use Criterion's OAS prior to service provider engagement;
- Use Criterion's OAS prior to service provider implementation;
- Request changes to be made by Criterion where appropriate, rather than directly with multiple service providers;
- Use the hypermedia links where provided, rather than statically storing URLs.

2 OPENAPI DEFINITION

OpenAPI definitions within Criterion Standards (as specified in accordance with the OpenAPI Specification https://swagger.io/docs/specification/about/) provide a machine-readable API definition, allowing quicker service implementations. OpenAPI definitions consist of the components detailed in the table below. The "prescriptive" column indicates if the content supplied by Criterion for this Standard is prescriptive or not.

OPENAPI COMPONENT	DETAILS	PRESCRIPTIVE
openapi	The OpenAPI specification version used. See https://swagger.io/specification/#appendix-a-revision-history	Yes
info	High-level description of the Criterion Standard: • Title; • Description; • version (specified in Criterion Standards Versioning Policy terms, e.g. v1.0.)	Yes
security	The security scheme for securing the service must be agreed between the trading partners and specified accordingly here.	No
paths	Contains the paths/operations available in the API: • defining individual endpoints in the API, and the HTTP methods supported by these endpoints; • includes all the details of the message exchange patterns and refers to the input/output data structures for each message; • defines the handled return code values (via RFC 7807); • can support semantic versioning (https://semver.org/). The service provider should remove any endpoints they do not wish to implement. The service provider should produce specific error messages including when functionality that is not supported is used. The service provider should consult with Criterion before extending into new end points, to aid in a universal approach.	No
server	The service provider must specify the API server and base URL. More than one server can be defined, for example one for production and one for a sandbox server. All paths are relative to server URLs. This cannot be prescriptive for obvious reasons. e.g. https://api.serviceprovider.com/	No
components/ schemas	Defines common data structures (schemas) referred to elsewhere in the API definition e.g., the paths/operations can refer to schema components for definitions of the message structures.	Yes
components/ securitySchemes	Defines common security schemes that can be used by the API's operations.	No

OpenAPI components that are NOT prescriptive can be customised for the specific requirements of an implementation. The API definition supplied as part of the Criterion Standard can be used as a basis for implementing a service conforming to the Criterion Standard. There will be no specific entries relating to securing the service or semantic versioning of the service operations/paths.

Trading partner specific HTTP headers can be supplied as part of the customisation of the OpenAPI definition.

3 END POINT VERSIONING

There are two distinct versions that need to be captured in a specification, the Criterion Standard version and the implementer's service version. The implementor version should be specified in the URL and the Criterion version in the header.
Criterion will version both the OAS and each contained end point.

4 SECURITY

Criterion is not prescriptive about security, as companies usually want to handle security in their own way. As the security landscape is constantly evolving, so it is better not to have security details baked into the Standard. Therefore, trading partners should ensure they have implemented adequate security, and include the security details in their OAS copy.
REST ids should be used to ensure no identifiable data is included in the URL. For example, instead of directly using a policy number as a URL parameter, use a GUID which would have no external meaning. Note that these IDs could be formed using a lookup table, or an encryption of a recognisable identifier.

5 EXTENSIBILITY

Criterion recognise the need for trading partners to exchange data which is not included in the OpenAPI Spec. To facilitate this, two options are provided as described below.

5.1 LINKS

For RESTful APIs where significant chunks of data need to be sent, a HATEOAS (Hypermedia As The Engine of Application State) approach can be used whereby one or more links are provided to handle extra data.
For more details on this, please see the REST guide which is available to users of Criterion REST Standards on the Criterion website.

5.2 TRADING PARTNER SPECIFIC DATA

Where HATEOAS is not suitable, for example where small bits of data are required throughout the message, or the data is contained inside an array, then trading partner specific data containers (called tpsData) can be used.

By using a named tpsData container, it is obvious where trading partner specific data is being deployed. Trading partners can use this extensibility feature in order to exchange information which is sensitive in competitive terms, or which has not been supported in the definition of this version of the Standard yet.

Some specifications already have tpsData objects included as optional objects spread throughout the business data payload of the request and response messages. However the agreed approach for future specifications is that tpsData will not be included by Criterion. Users of Standards are welcome to insert tpsData objects as required if not already present. These tpsData containers should appear as the last property in a particular object definition.

5.2.1 HOW TO USE TPSDATA

Within tpsData there should be a hierarchy, with a root object identifying the sender domain of the tpsData, and the payload contained under that. This facilitates handling similar tpsData from multiple trading partners, as they can be separated out in the specification copy under different hierarchies.

This allows receivers to combine multiple tpsData into a single specification so that tpsData processing can be branched off anywhere in the process flow rather than up front at ingestion, relieving the need for multiple specifications.

5.2.2 EXAMPLE INSTANCE

This example instance shows tpsData included in two places. This represents an example of how an adviser request message could look. In this case the request comes from best-advisers.co.uk.

Trading partners should update their copy of the Standard specification to define the expected payload within tpsData.
Note the use of an object with the name 'best-advisers.co.uk' in the example below. It is proposed that all tpsData payload is provided within an object which identifies the sender. As shown in the subsequent example, the specification copy would be updated with multiple such hierarchies in order to define expected data from multiple different trading partners.


```
{
  "requestor_reference": "abc123",
  "current_product_provider": "Acme Provider",
  "adviser": {
    "full_name": "John Doe",
    "adviser_reference_number": "1a2b3c",
    "adviser_issuing_authority_name": "FCA"
  },
  "policyholder": {
    "person": {
      "given_names": ["Jane"],
      "family_name": "Smith"
    },
    "tpsData": {
      "best-advisers.co.uk": {
        "policyholderProfile": "Novice",
        "commsPreference": "email",
        "previousAuthExists": true
      }
    }
  },
  "include_specified_contracts_only_ind": true,
  "authorisations": [
    {
      "authorisation_type": "HandWrittenScanned",
      "image_file_format": "PNG",
      "base64_binary": "asdjfay74385uk4tkjklsdjg"
```

```
    }
  ],
    "tpsData": {
      "best-advisers.co.uk": {
        "urgencyLevel": "high",
        "riskRating": "3",
        "validTo": "2025-05-01",
        "authorisedProcesses": [
          "PropertySell",
          "PropertyValuation",
          "Admin"
        ]
      }
    }
}
```

5.2.3 EXAMPLE OAS

This example OAS excerpt shows how the tpsData identifier can be used to distinguish similar data from multiple trading partners. This represents an example of how a provider specification copy could look. The original Criterion specification would not have tpsData defined.

In this case a provider defines data structures for tpsData received from three different advisers. Notice in the first tpsData below that *urgencyLevel* is defined as a *string* for best-advisers.co.uk, whereas it is defined as an *integer* for another-adviser.org.uk. This illustrates how adding an identifying object adds flexibility, as the provider can accept different tpsData payloads from different trading partners and validate them all against their copy of the specification.

```
RequestBody:
  type: object
  additionalProperties: false
  properties:
    requestor_reference:
      type: string
    current_product_provider:
      type: string
    adviser:
      $ref: "#/components/schemas/Adviser"
    policyholder:
      $ref: "#/components/schemas/Policyholder"
    include_specified_contracts_only_ind:
      type: boolean
    authorisations:
      type: array
      minItems: 1
      maxItems: 100
      items:
```

```
        $ref: '#/components/schemas/Authorisation'
    tpsData:
      type: object
      additionalProperties: false
      properties:
        best-advisers.co.uk:
          type: object
          additionalProperties: false
          properties:
            urgencyLevel:
              type: string
            riskRating:
              type: integer
            validTo:
              type: number
              format: date
            authorisedProcesses:
              type: array
              minItems: 1
              maxItems: 100
              items:
                type: string
                enum:
                  - Admin
                  - PropertySell
                  - PropertyValuation
        another-adviser.org.uk:
          type: object
          additionalProperties: false
          properties:
            urgencyLevel:
              type: integer
            risky:
              type: boolean
            validTo:
              type: number
              format: date
            authorisedProcesses:
              type: array
              minItems: 1
              maxItems: 2
              items:
                type: string
                enum:
                  - Admin
                  - All
                  - Restricted
Adviser:
  type: object
```

```
  additionalProperties: false
  properties:
    full_name:
      type: string
    adviser_reference_number:
      type: string
    adviser_issuing_authority_name:
      type: string
Policyholder:
  type: object
  additionalProperties: false
  properties:
    person:
      $ref: '#/components/schemas/Person'
    tpsData:
      type: object
      additionalProperties: false
      properties:
        best-advisers.co.uk:
          type: object
          additionalProperties: false
          properties:
            policyholderProfile:
              type: string
            commsPreference:
              type: string
            previousAuthExists:
              type: boolean
        ace-advice.co.uk:
          type: object
          additionalProperties: false
          properties:
            disclosureComplete:
              type: boolean
Person:
  type: object
  additionalProperties: false
  properties:
    given_names:
      type: array
      minItems: 1
      maxItems: 10
      items:
        type: string
    family_name:
      type: string
Authorisation:
  - $ref: '#/components/schemas/HandWrittenScanned'
HandWrittenScanned:
```

```yaml
type: object
additionalProperties: false
properties:
  authorisation_type:
    type: string
    enum:
      - HandWrittenScanned
  image_file_format:
    type: string
    enum:
      - PNG
      - JPEG
  base64_binary:
    minLength: 1
    maxLength: 7000000
```

6 GENERAL NOTES

6.1 EMPTY DATA ITEMS

The specification validation rules ensure that no empty data items are sent.