

JSON, JSON Schema, and OpenAPI

Octavian Nadolu, Syncro Soft
octavian.nadolu@oxygenxml.com
@OctavianNadolu

© 2022 Syncro Soft SRL. All rights reserved.

Agenda

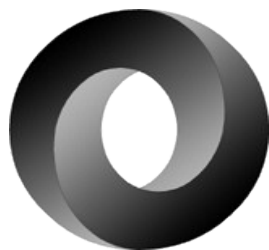
- Smart editing of JSON documents based on JSON Schema
- JSON Schema editing in Design mode
- Specialized JSON Tools
- OpenAPI, AsyncAPI, JSON-LD support



JSON

(JavaScript Object Notation)

- Data representation format
- Used for API and Configs
- Lightweight and easy to read/write



<http://json.org>

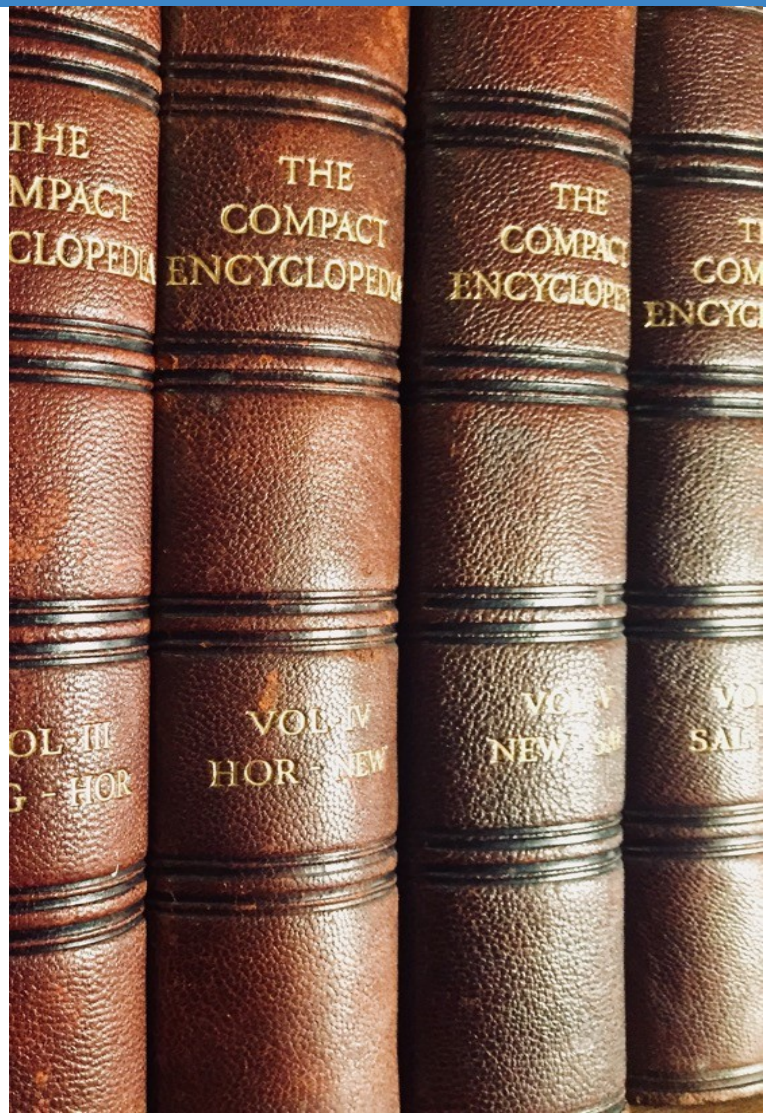


JSON Schema

JSON Schema is a vocabulary that allows you to **annotate** and **validate** JSON documents



<http://json-schema.org>



JSON and JSON Schema Support in Oxygen

- **Edit JSON** based on JSON Schema
- **Validate JSON** against JSON Schema
- **JSON Schema Editor** - specialized editor with various editing features
- Tools
 - Generate JSON **Schema Documentation**
 - **Generate Sample JSON** Files from a JSON Schema
 - **Generate JSON Schema** from a JSON File
 - **XSD to JSON** Schema Converter

Editing JSON

- JSON Text, Grid, Author Editing Mode
- JSON Outline View
- Validation and Content Completion Assistant based on JSON Schema



Text Editing Mode

- Syntax Highlights
- Structure Folding
- Format and Indent
- New Document Templates

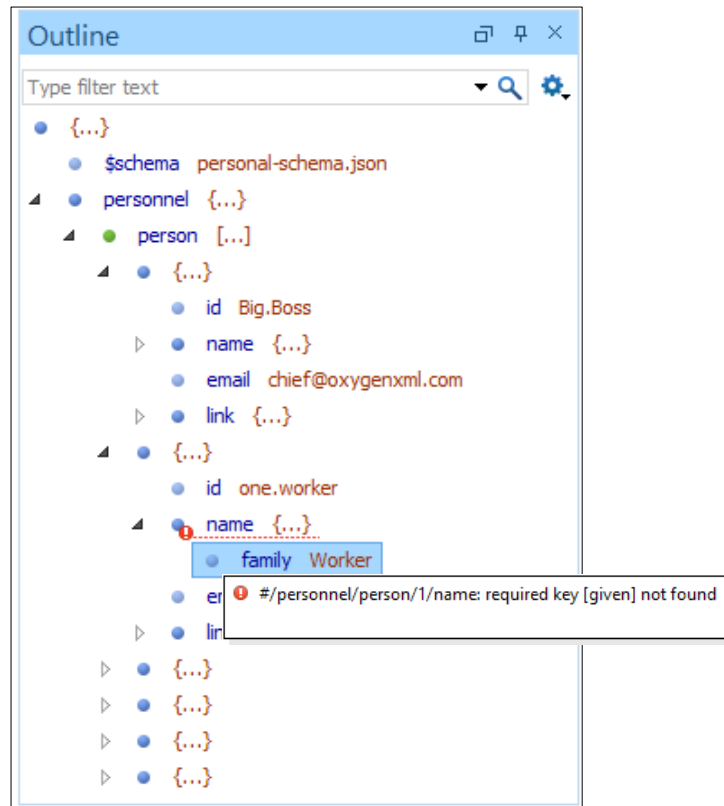
A screenshot of the Oxygen XML Studio interface showing a JSON file named "personal.json" in text editing mode. The editor displays a JSON object with syntax highlighting and structure folding. The JSON content is as follows:

```
1 {
2   "$schema": "personal-schema.json",
3   "personnel": {
4     "person": [
5       {
6         "id": "Big.Boss",
7         "name": {
8           "family": "Boss",
9           "given": "Big"
10        },
11        "email": "chief@oxygenxml.com",
12        "link": {
13          "subordinates": [
14            "one.worker",
15            "two.worker",
16            "three.worker",
17            "four.worker",
18            "five.worker"
19          ]
20        }
21      }
22    ]
23  }
24 }
```

The interface includes a tab for "personal.json", a line number gutter on the left, and a scrollbar on the right.

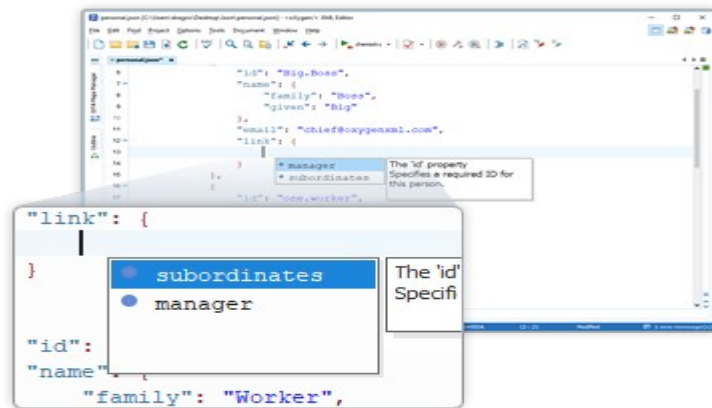
JSON Outline View

- Document Overview
- Synchronization with Editor
- Filtering
- Error Markers



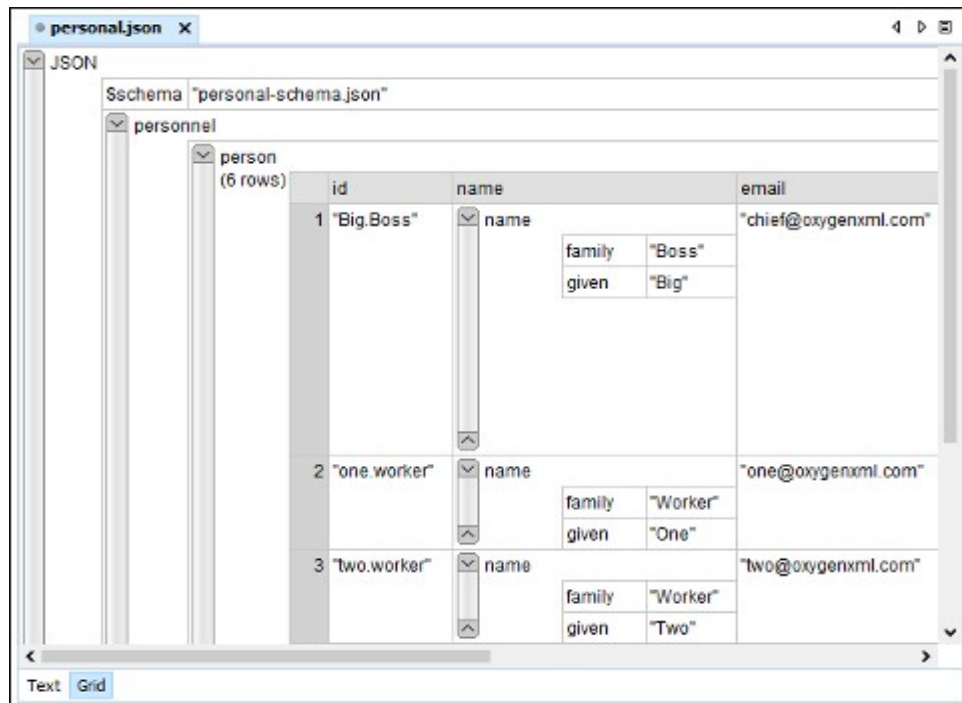
Content Completion

- Context-Sensitive
- Valid Proposals
- Documentation from Schema
- Code Templates



Grid Editing Mode

- JSON content is represented in Grid mode
- JSON data and structure can be easily manipulated
- Table-specific operations or drag-and-drop operations



JSON Visual Author Editing Mode

- JSON document is opened in Author mode
- Create your own JSON framework
- Customize using CSS

The screenshot shows the Oxygen XML Editor interface in Author mode. The main window displays a JSON document titled "residentCardForm.json" rendered as a web form. The form is titled "Application to Replace Permanent Resident Card" and is divided into two parts:

Part 1. Information About You

- 1. Gender: Male Female
- 2. Class of Admission: T1 - 1ST PREF SELECTED ALIEN (dropdown)
- 3. Date of Birth (mm/dd/yyyy): 01/11/1984 (calendar icon)
- 4. Date of Admission (mm/dd/yyyy): 01/02/2012 (calendar icon)
- 5. City/Town/Village of Birth: Prague
- 6. U.S. Social Security number (if any):
- 7. Country of Birth: Czech Republic (dropdown)

Part 2. Application Type

NOTE: If your conditional status is expiring within the next 90 days, then do not file this application.

My status is (Select only one box):

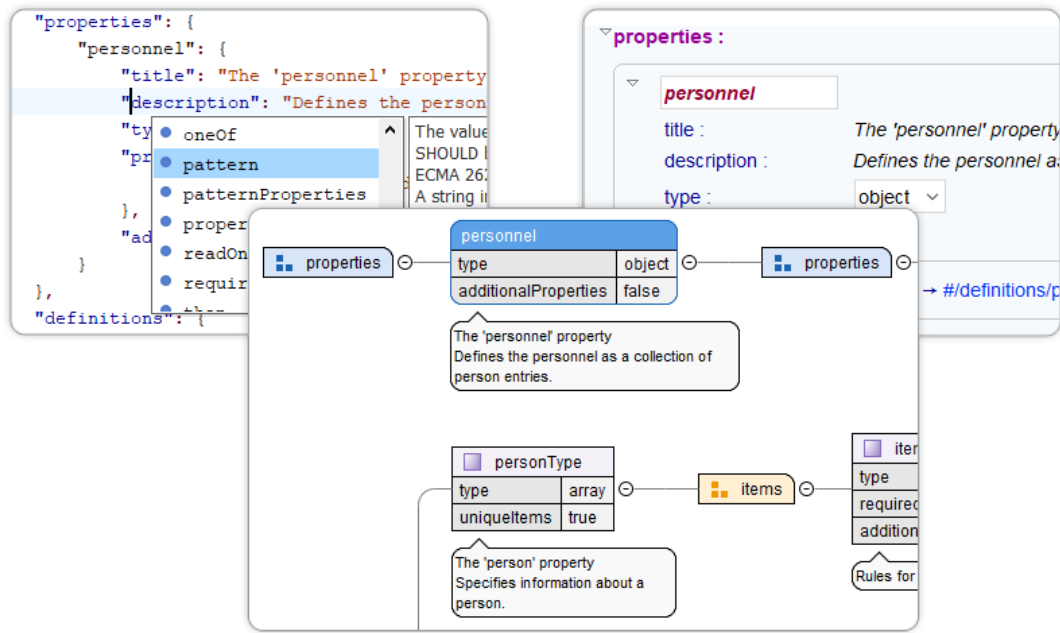
- 1.a. Permanent Resident
- 1.b. Permanent Resident - In Commuter Status
- 1.c. Conditional Permanent Resident

The bottom of the editor shows the "Author" mode selected in the "Text Grid Author" tabs.

JSON Schema Editor

Design, develop, and edit JSON Schemas in:

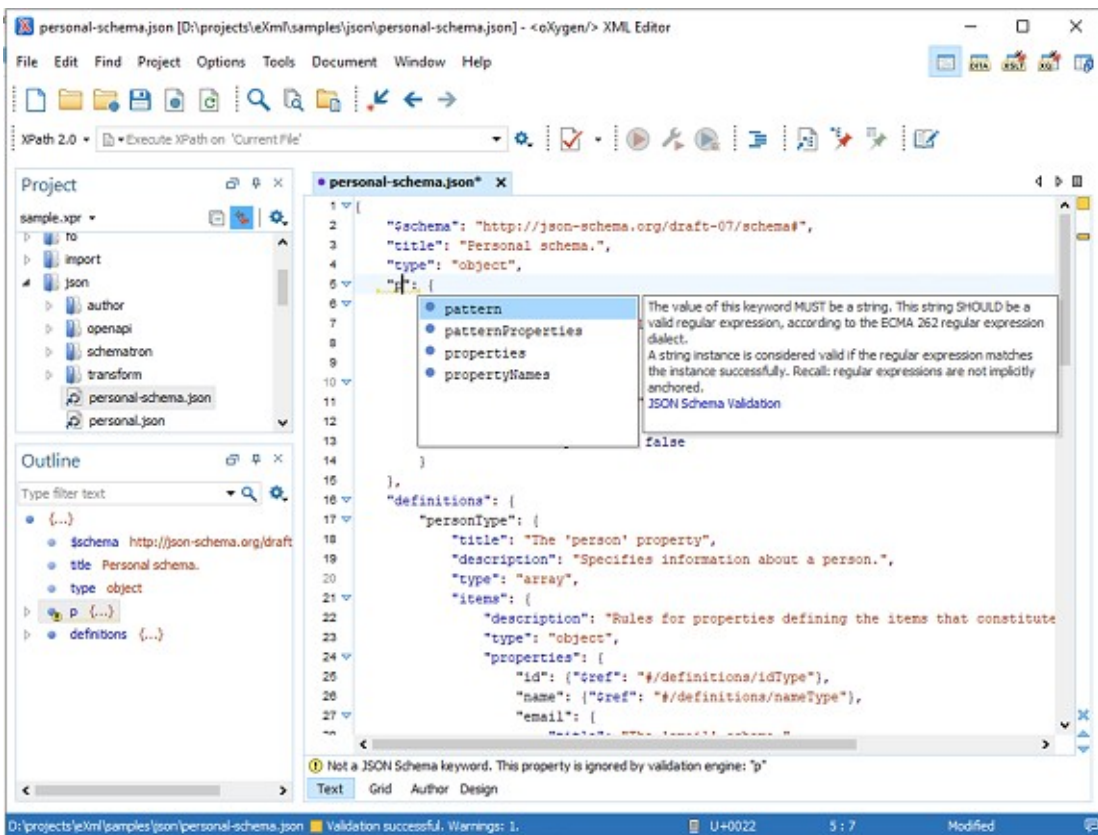
- Text Editing Mode
- Author Editing Mode
- Schema Design Mode



Text Editing Mode

Text editing mode is packed full of editing helpers

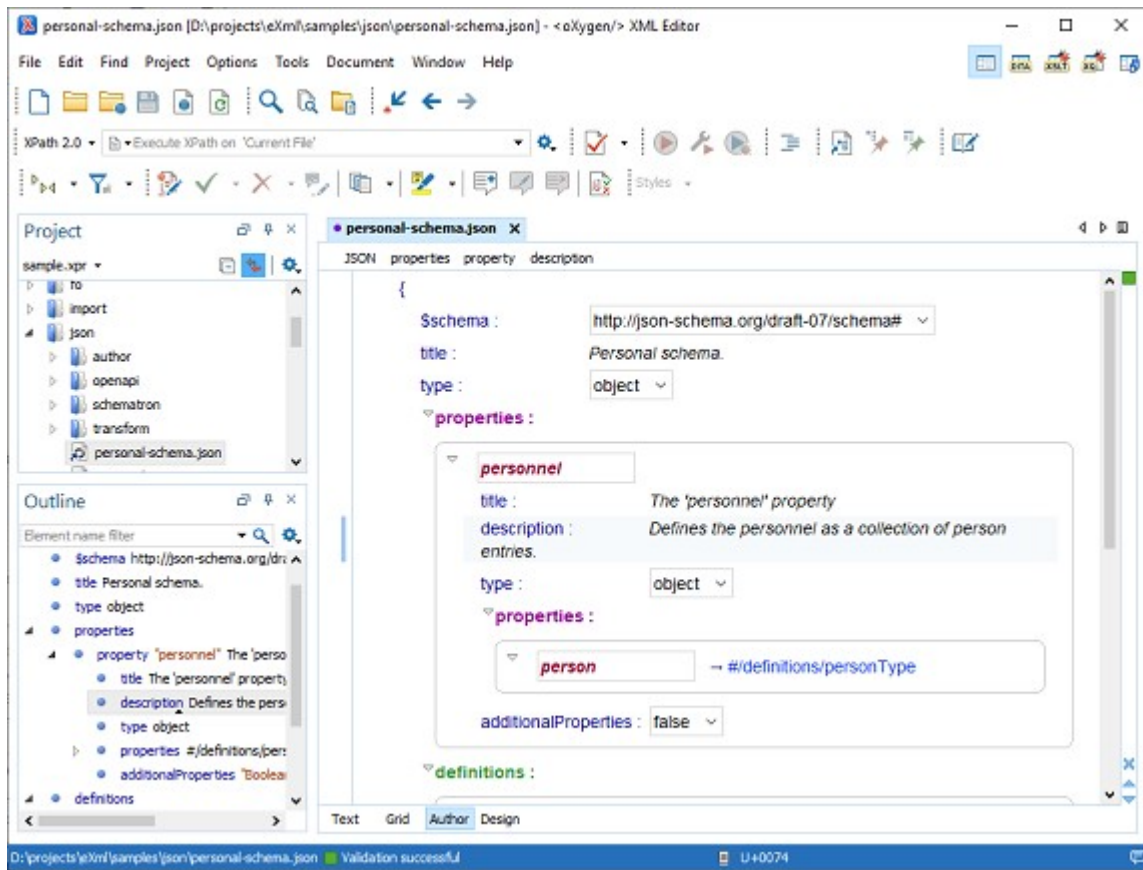
- JSON **Outline View**
- JSON-specific **Syntax Highlighting**
- Search and **Find/Replace**
- **Drag and Drop**
- **Validation**
- **Format and Indent (Pretty Print)**



Author Editing Mode

Visual editing mode for JSON Schema documents:

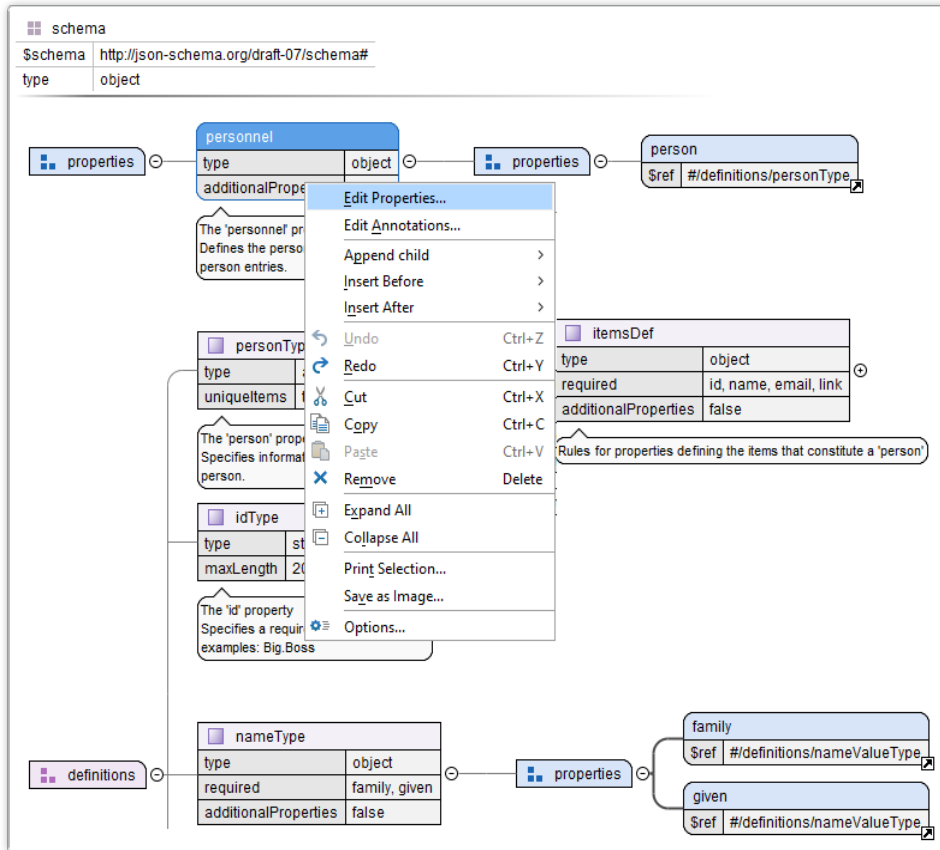
- JSON Schema **framework**
- **Content completion** support
- **Validation**
- Specific **CSS** for rendering
- Create your own **custom JSON framework**



Schema Design Mode

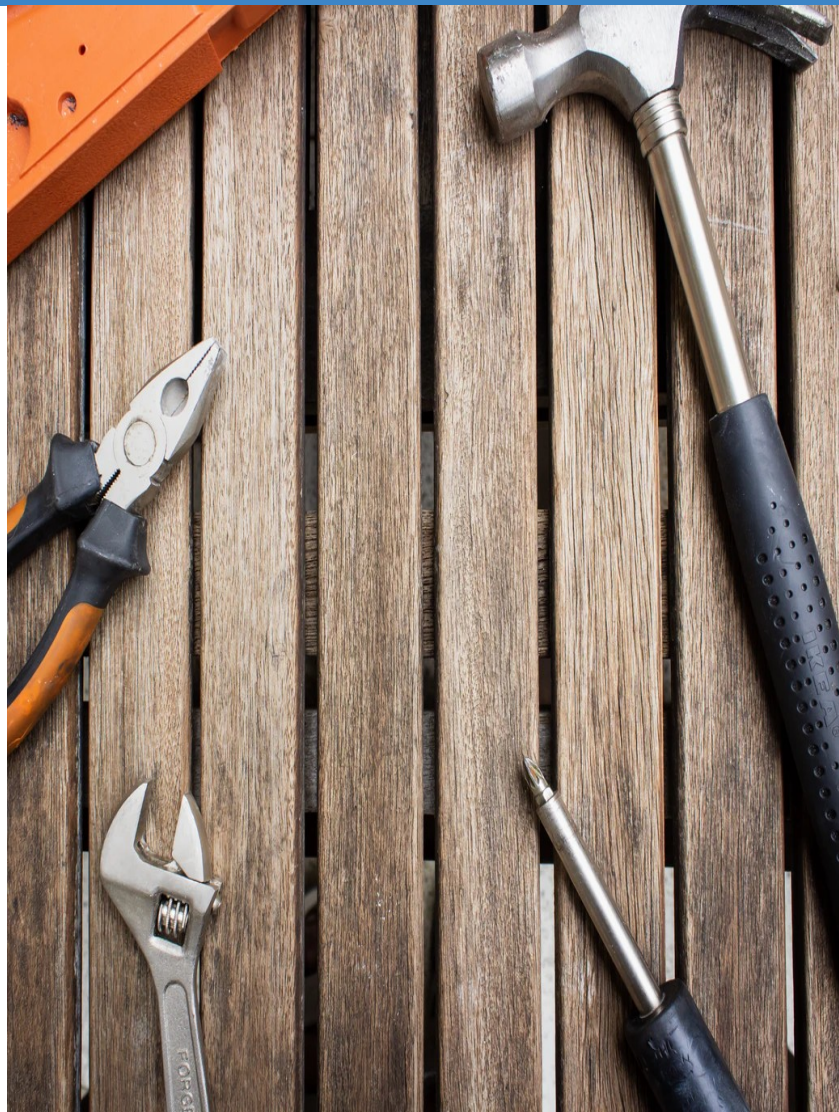
Visualize, edit, and understand JSON Schemas

- In-Place Component Editing
- Edit Properties
- Edit Annotations
- Schema Editing Actions
- Move Component Up/Down Actions
- Print/Save as Image



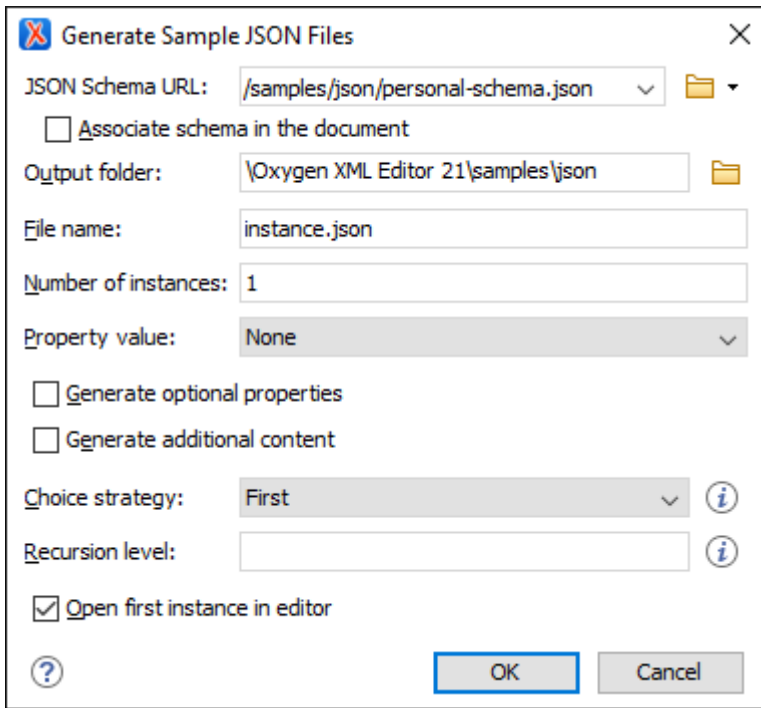
JSON Tools

- Convert between **JSON and XML**
- Convert between **JSON and YAML**
- Generate **sample JSON files** from JSON Schema
- **Generate JSON Schema** from a JSON instance
- **XSD to JSON Schema** converter
- JSON Schema **documentation**



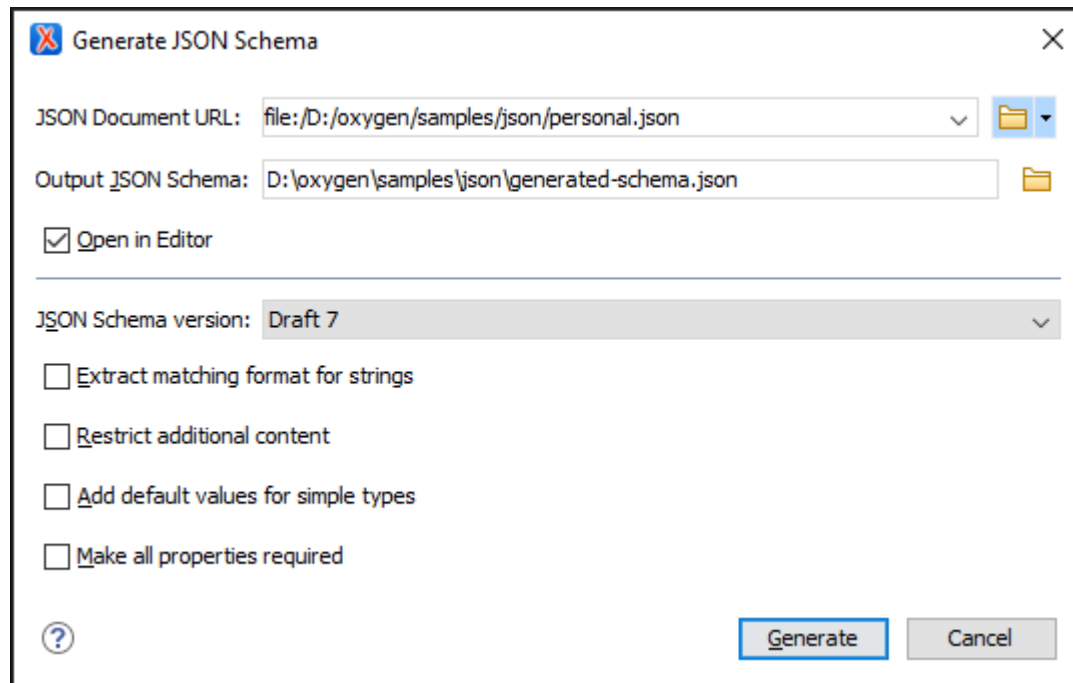
Instance generator from a JSON Schema

- Tool for generating sample JSON files from a JSON Schema



Generate JSON Schema from JSON

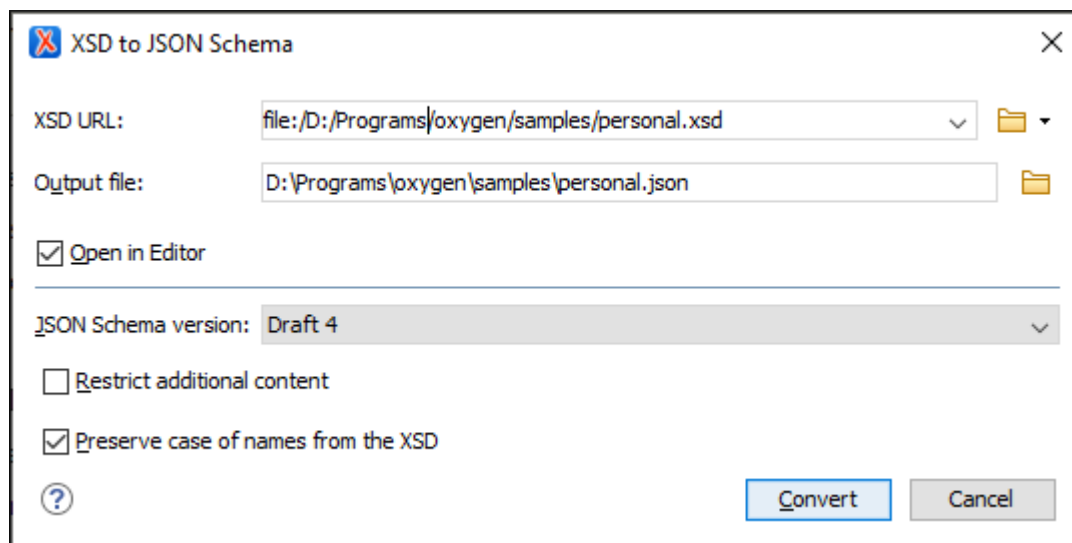
- Tool for generating JSON Schema



oxygenxml.com/doc/ug-editor/topics/json-schema-instance-generator.html

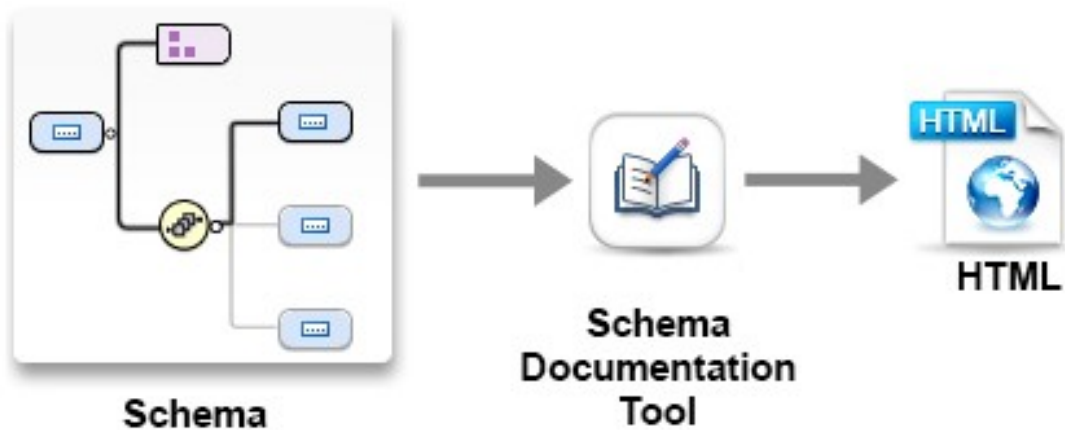
XSD to JSON Schema converter

- Tool for converting an XML Schema file (XSD) to a JSON Schema



Generate JSON Schema Documentation

- Tool for generating detailed documentation for a JSON Schema file in HTML format



JSON Schema Documentation

- Generate documentation in one file or split into multiple files
- Option to include components details
- Display the diagram image for each component

The screenshot shows a web browser window titled "Documentation for UBL-Order-2". The main content area displays the "Property Order" details. On the left, there are expandable sections for "Annotations", "Diagram", "Type", "Constraints", "Array Items", "Additional Items", "Used by", and "Source". The "Diagram" section shows a visual representation of the schema: an "Order" object with properties "maxItems": 1, "minItems": 1, and "type": "array". It is connected to an "items" array, which in turn points to an "ItemsDef" object with a "\$ref": "#/definitions/Order".

The "Annotations" section contains a "Description": "A document used to order goods and services."

The "Type" section shows "array".

The "Constraints" section shows "Unique Items: false".

The "Array Items" section shows "Items" and a link to "UBL-Order-2.1.json.html#/properties/Order/items".

The "Additional Items" section shows "true".

The "Used by" section shows "Schema" and "#/schema".

The "Source" section shows a JSON snippet:

```

"Order": {
  "type": "array",
  "minItems": 1,
  "maxItems": 1,
  "description": "A document used to order goods and services.",
  "items": {"$ref": "#/definitions/Order"}
}
    
```

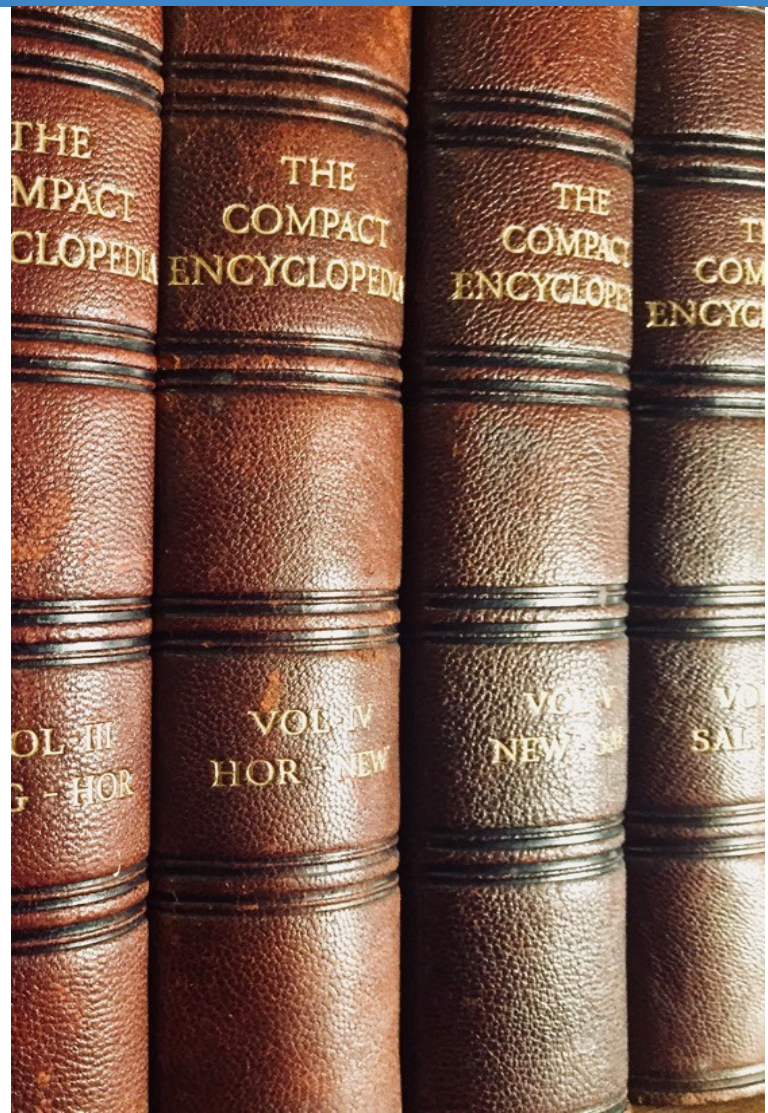
On the right side, there is a "Showing:" panel with checkboxes for "Annotations", "Diagram", "Properties", "Constraints", "Used By", and "Source", all of which are checked. A "Close" button is located below the checkboxes.

OpenAPI Specification(OAS)

- Is a format for describing and documenting APIs
- Allows both humans and computers to discover and understand the capabilities of a service



www.openapis.org



OpenAPI Documents

- Can be written in JSON or YAML
- Describes the inputs and outputs of an API
- Include information such as:
 - where the API is hosted
 - what authorization is required to access it
 - details for consumers and producers

OpenAPI Editor

- Syntax highlights
- Validation conforms with the schema
- Content completion based on the schema
- Outline view
- Search and replace

```
1 {
2   "openapi": "3.0.0",
3   "info": {
4     "version": "1.0.0",
5     "title": "Petstore",
6     "description": "A sample API that uses a petstore as an example to de
7     "termsOfService": "http://swagger.io/terms/",
8     "contact": {
9       "name": "Swagger API Team",
10      "email": "apiteam@swagger.io",
11      "url": "http://swagger.io"
12    },
13    "license": {
14      "name": "Apache 2.0",
15      "url": "https://www.apache.org/licenses/LICENSE-2.0.html"
16    },
17    "$comment": "This sample file's original source is available at OpenA
18  },
19  "server": [
20    {
21      "url": "http://petstore.swagger.io/api"
22    }
23  ],
24  "paths": {
25    "/pets": {
26      "description": "Returns all pets from the system that the use
27      "operationId": "findPets",
28      "parameters": [
29        {
30          "name": "tags",
31          "in": "query",
32          "description": "tags to filter by",
33          "required": false,
```

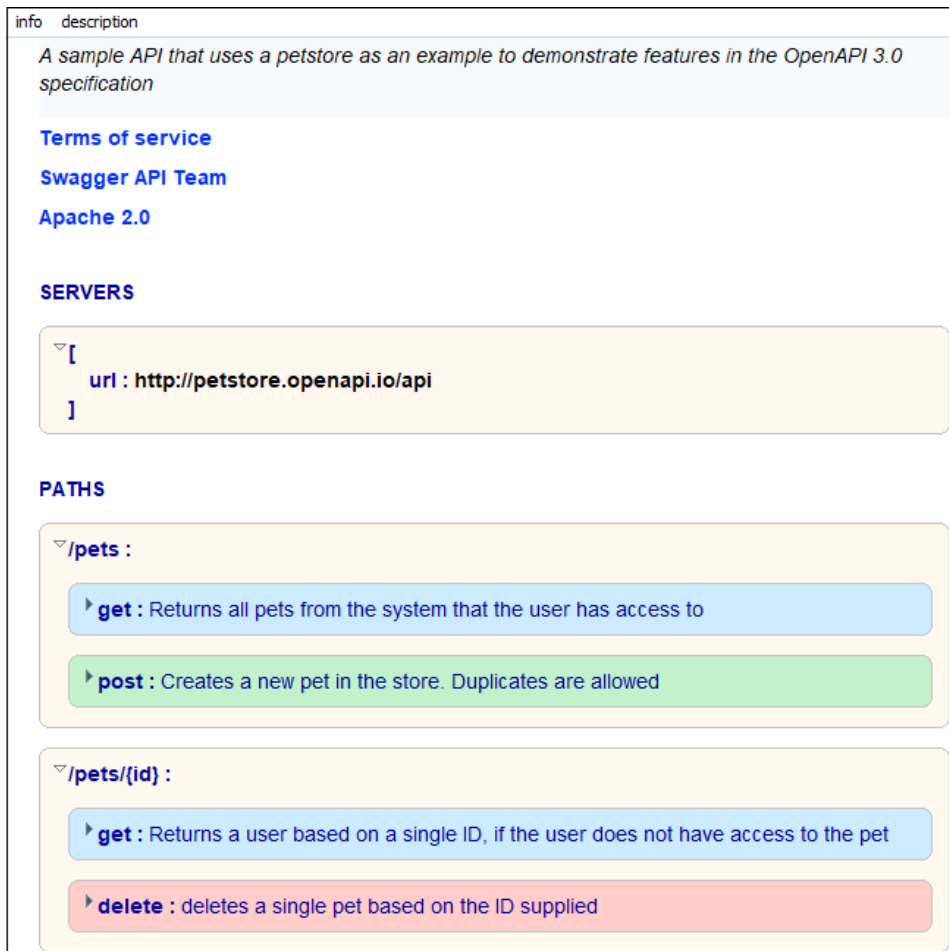
externalDocs
security
servers
tags

#/server: extraneous key [server] is not permitted

Text Grid Author

OpenAPI Visual Editor

- Edit documentation
- Form controls to change values
- Collapsible sections
- Customize interface with CSS



The screenshot displays the OpenAPI Visual Editor interface for a sample API. The interface is organized into several sections:

- info description:** A sample API that uses a petstore as an example to demonstrate features in the OpenAPI 3.0 specification.
- Terms of service:** Includes links for [Terms of service](#), [Swagger API Team](#), and [Apache 2.0](#).
- SERVERS:** A collapsible section containing a single server configuration:

```
{
  url : http://petstore.openapi.io/api
}
```
- PATHS:** A collapsible section containing two path definitions:
 - /pets :** Contains two methods:
 - get :** Returns all pets from the system that the user has access to (blue background).
 - post :** Creates a new pet in the store. Duplicates are allowed (green background).
 - /pets/{id} :** Contains two methods:
 - get :** Returns a user based on a single ID, if the user does not have access to the pet (blue background).
 - delete :** deletes a single pet based on the ID supplied (red background).

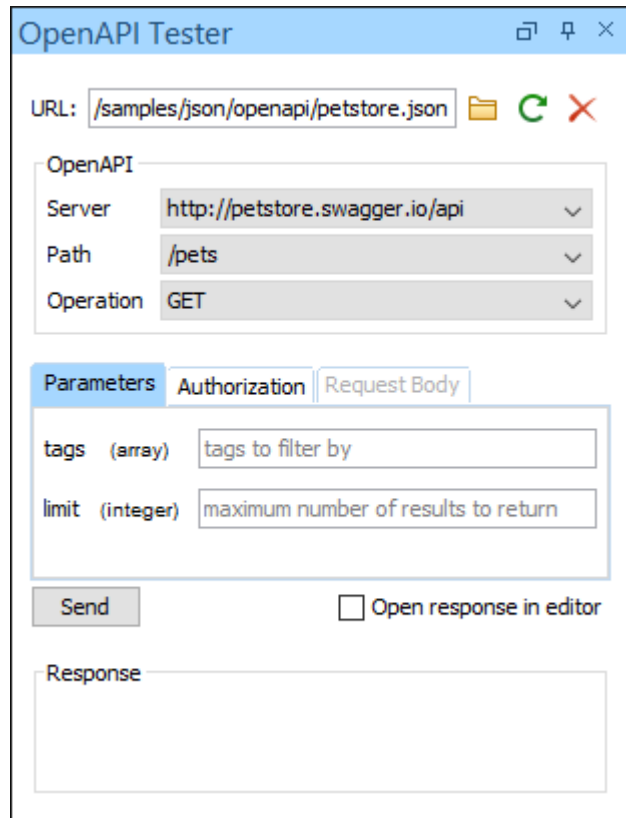
OpenAPI Documentation

- Tool for generating detailed documentation for a OpenAPI documents in HTML format



OpenAPI Tester Tool

- GUI interface testing tool for OpenAPI files
- Execute API requests and validate responses on the fly
- OpenAPI files in JSON or YAML format



The screenshot shows the OpenAPI Tester application window. At the top, the URL is set to `/samples/json/openapi/petstore.json`. Below this, the OpenAPI configuration is shown with three dropdown menus: Server (http://petstore.swagger.io/api), Path (/pets), and Operation (GET). The Parameters tab is active, showing two input fields: 'tags (array)' with the value 'tags to filter by' and 'limit (integer)' with the value 'maximum number of results to return'. There is a 'Send' button and a checkbox for 'Open response in editor'. At the bottom, there is a large text area labeled 'Response'.

Other APIs

- AsyncAPI - defines the interfaces of asynchronous APIs
- JSON-LD - is a lightweight Linked Data format



Conclusion

- Edit and validate JSON documents
- Edit/Design JSON Schema documents
- JSON/JSON Schema Tools
- Edit, validate, generate documentation, and test OpenAPI



Future Plans

- Search and Refactor in JSON **Diagram**
- Better JSON Schema **Documentation**
- **Quick fixes** for JSON problems
- Improve **OpenAPI** support
- Content completion in **YAML**



Resources

- oxygenxml.com/json.html
- oxygenxml.com/openapi.html
- oxygenxml.com/doc/ug-editor/topics/editing-JSON.html
- oxygenxml.com/doc/ug-editor/topics/json-tools.html
- <https://www.json.org>
- <https://json-schema.org/>
- <https://www.openapis.org>
- <https://www.asyncapi.com>
- w3.org/TR/json-ld11/



Video Demos

- oxygenxml.com/demo/json_schema_palette.html
- oxygenxml.com/demo/introducing_the_json_schema_design.html
- oxygenxml.com/demo/json_author.html
- oxygenxml.com/demo/json_tools.html
- oxygenxml.com/demo/json_validation.html
- oxygenxml.com/demo/json_editing.html
- oxygenxml.com/demo/json_query.html



Questions?

Octavian Nadolu
Product Manager at Syncro Soft

octavian.nadolu@oxygenxml.com

Twitter: [@OctavianNadolu](https://twitter.com/OctavianNadolu)

LinkedIn: [octaviannadolu](https://www.linkedin.com/in/octaviannadolu)