

# Table of Contents

JADN Sandbox Capabilities

[Schema Creation](#)

[How to get started...](#)

[Schema Visualization](#)

[Schema Translation](#)

[Data Creation](#)

[Data Validation](#)

[Example Data Generation](#)

[Schema Transformation](#)

[Other Features](#)

Docker Desktop

[How to Get the JADN Sandbox Image](#)

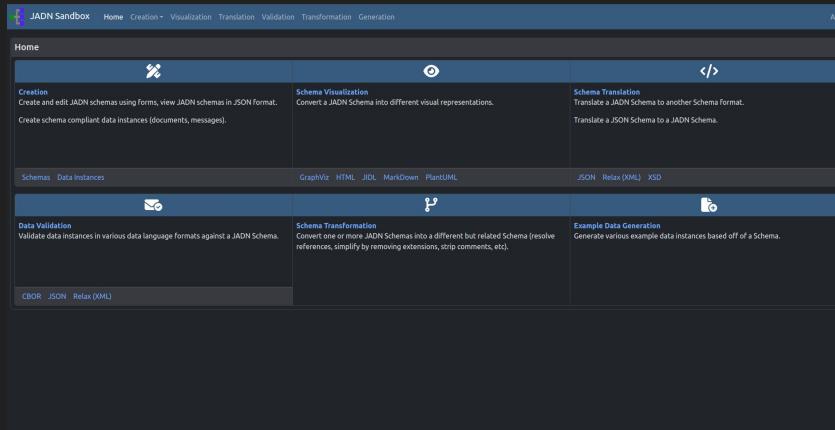
[How to Run the JADN Sandbox Image](#)

[How to Update the JADN Sandbox Image](#)

[How to Stop the JADN Sandbox Image](#)

[How to Learn More about Docker](#)

# JADN Sandbox Capabilities



# JADN Sandbox Table of Contents

## Schema Creation

Feature: Editor Style

To Start

Add Data

Edit Data

Feature: Types Outline

More Functions and Features

## Schema Transformation

Result: A Resolved Schema

Result: Strip Comments

## Other Features

Theme Switcher

Add Custom Files

Remove Custom Files

Download Files

## How to get started...

Schema Visualization

Results: Selecting One Language

Results: Selecting Multiple Languages

Schema Translation

Data Creation

View JSON/Creator

Data Validation

Invalid Results

Valid Results

Example Data Generation

Results

# Schema Creation

Attempt to create valid JADN Schemas

JADN Sandbox Home Creation Visualization Translation Validation Transformation Generation About

Home Schema Creation Data Creation

**Creation**  
Create and edit JADN schemas using forms, view JADN schemas in JSON format.  
Create schema compliant data instances (documents, messages).

**Schema Visualization**  
Convert a JADN Schema into different visual representations.

**Schema Translation**  
Translate a JADN Schema to another Schema format.  
Translate a JSON Schema to a JADN Schema.

Schemas Data Instances GraphViz HTML JIDL MarkDown PlantUML JSON Relax (XML) XSD

**Data Validation**  
Validate data instances in various data language formats against a JADN Schema.  
CBOR JSON Relax (XML)

**Schema Transformation**  
Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc.).

**Example Data Generation**  
Generate various example data instances based off of a Schema.

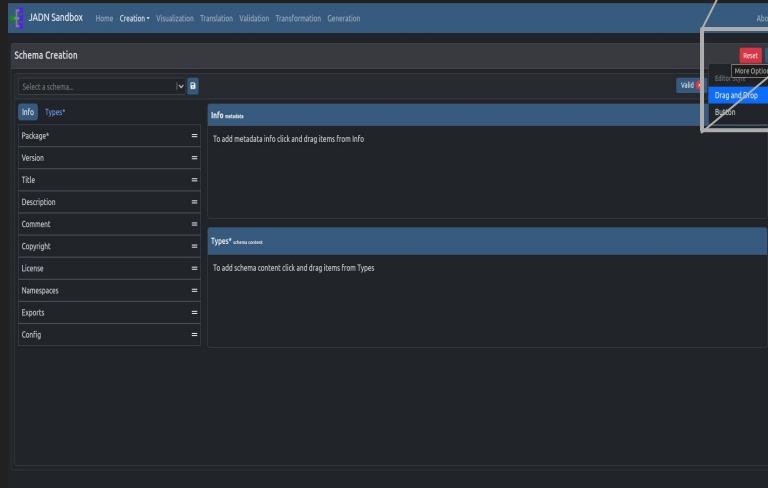
localhost:8082/create/schema v0.10.0\_1705587211962

Go to: Creation  
When the drop down menu appears, select Schema Creation

# Feature: Editor Style

Choose your schema creation editor style:

- Drag and Drop (default)
- Button



Drag and drop style (default)

This screenshot shows the same JADN Sandbox Schema Creation interface as the previous one, but with the 'Button' editor style selected. The 'Drag and Drop' button is now grayed out, and a new red button labeled 'Button' is highlighted. A tooltip above the red button also displays the 'Button' option. The rest of the interface remains the same, with its dark-themed design.

Button style

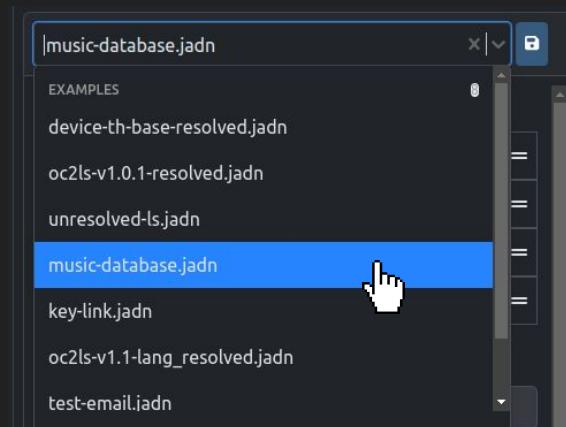
# To start...

- 1 Select an example schema  
or
- 2 Upload a syntactically valid JADN Schema  
or
- 3 From scratch (blank schema)

The screenshot shows the JADN Sandbox interface with the 'Creation' tab selected. In the 'Schema Creation' section, there is a dropdown menu labeled 'Select a schema...' containing several schema files: 'oc2ls-v1.0.1-resolved.jadn', 'unresolved-ls.jadn', 'music-database.jadn' (which is currently selected), 'key-link.jadn', 'oc2ls-v1.1-lang\_resolved.jadn', 'test-email.jadn', and 'start-up-template.jadn'. Below the dropdown is a button labeled 'Upload Custom File ...'. To the right of the schema selection area, there are two panels: 'Info metadata' and 'Types\* schema content', both with placeholder text and instructions.

# Option 1 to Start: Select a Schema

Selecting a preloaded Schema from the drop down menu will automatically generate the Schema in the Schema Creator.



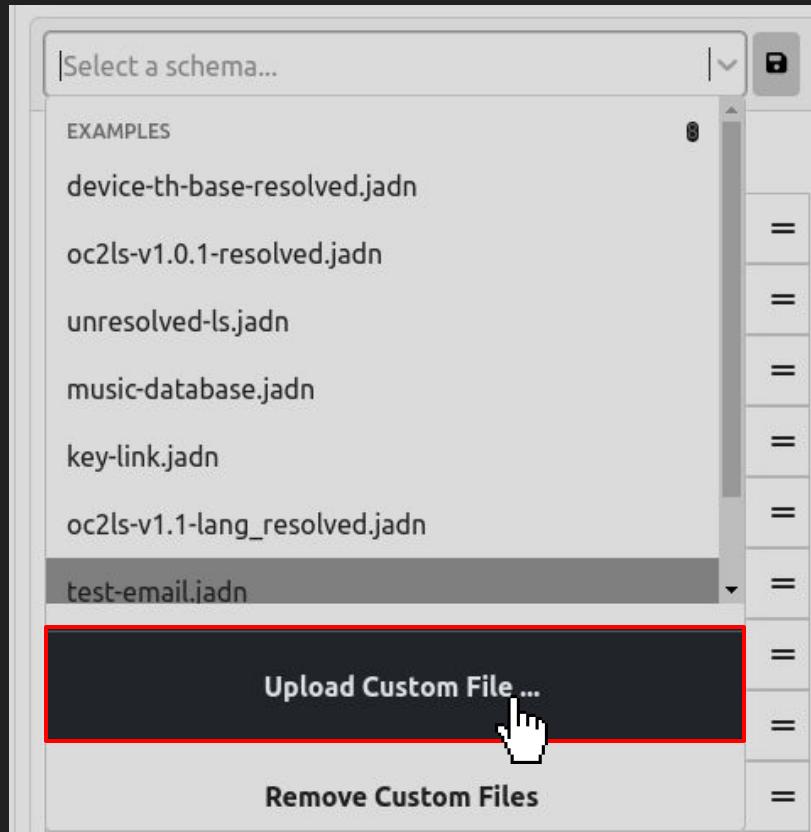
The screenshot shows the JADN editor interface with a valid JSON schema for a music database. The schema includes fields for package, version, title, description, and license. The description field contains a detailed explanation of the library's purpose. The schema is valid, as indicated by the green 'Valid' button at the top right.

```
music-database.jadn
{
  "Info": {
    "Comment": null,
    "Copyright": null,
    "Namespaces": null,
    "Config": null
  },
  "Info metadata": {
    "Package": "http://fake-audio.org/music-lib",
    "Version": "1.0",
    "Title": "Music Library",
    "Description": "This information model defines a library of audio tracks, organized by album",
    "License": "CC0-1.0"
  },
  "Exports": [
    "Library"
  ]
}
```

## Option ② to Start: Upload a Schema

Click ‘Upload Customer File...’ to upload a file using the computer’s file loader

This will upload any JADN file and then attempt to validate it.

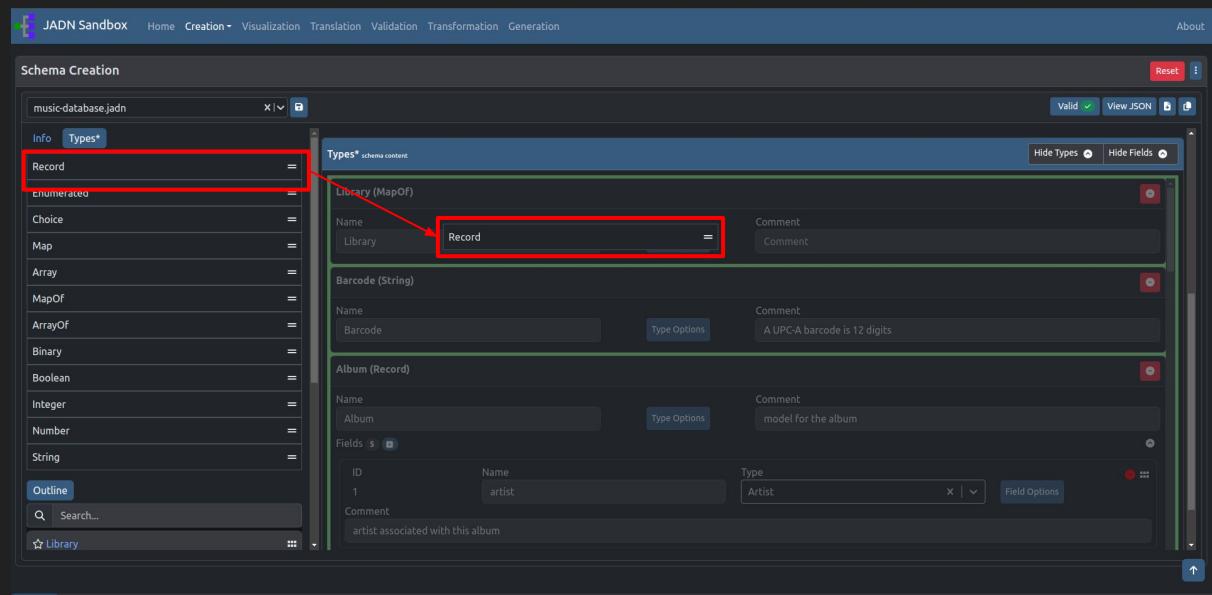


# Add Data: Drag and Drop Style

Using the default drag and drop style creator, drag items from left to right.

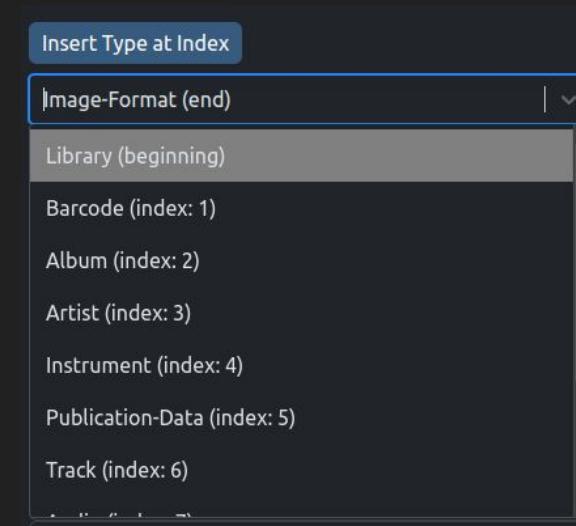
Drop the card when backgrounds are highlighted green.

When dragging, the background will change colors (in this case, to dark gray) to indicate where you can drop the card. It will then change to a green background when you are able to drop the card.

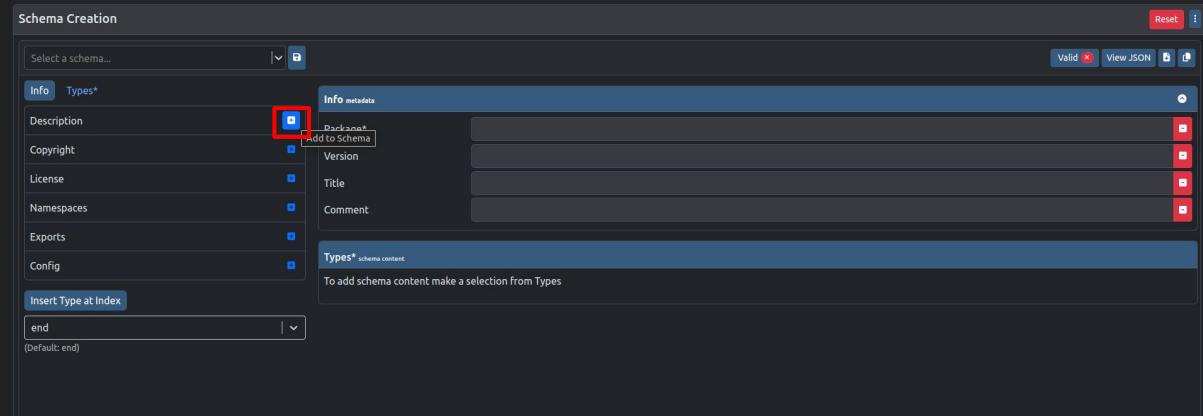


# Add Data: Button Style

The ‘Insert Type at Index’ allows you to select where you would like to add your Type. Types are added to the end of the Schema by default.



Click to add Info or Type to the schema.

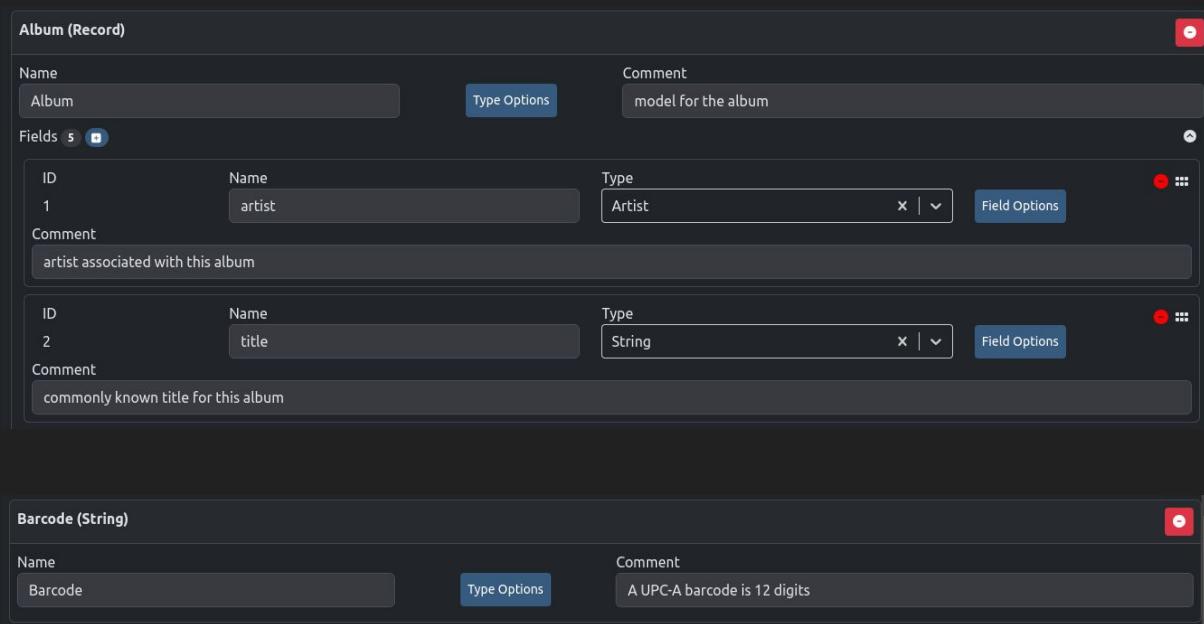


# Edit Data: Add, change, or remove Types

The cards dropped on the right can be edited by clicking within input fields or removed .

Fields, if applicable, can be:

- Added,
- removed,
- hidden, and
- easily rearranged.



The screenshot shows a data editor interface with three cards:

- Album (Record)**:
  - Name: **Album** (with **Type Options** button)
  - Comment: **model for the album**
  - Fields (5):
    - ID: 1, Name: **artist**, Type: **Artist** (with **Field Options** button)
    - Comment: **artist associated with this album**
- Barcode (String)**:
  - Name: **Barcode** (with **Type Options** button)
  - Comment: **A UPC-A barcode is 12 digits**
- Unnamed Card**:
  - ID: 2, Name: **title**, Type: **String** (with **Field Options** button)
  - Comment: **commonly known title for this album**

# Feature: Schema Types Outline

You can easily search or rearrange the Types within the Outline.



- ★ The cards within the outline can also be starred for future reference.

Clicking on the Type Name will take you to the card in the Types section of the schema on the right.

A screenshot of a software interface titled "Outline". It features a search bar at the top with the placeholder "Search...". Below the search bar is a list of schema types, each preceded by a star icon and a three-dot menu icon. The types listed are Library, Barcode, Album, Artist, Instrument, Publication-Data, Track, Audio, and Audio-Format. The "Album" type is currently selected, highlighted with a darker background.

A screenshot of a software interface titled "Outline". It features a search bar at the top with the placeholder "Search...". Below the search bar is a list of schema types, each preceded by a star icon and a three-dot menu icon. The types listed are Library, Barcode, Album, Artist, Instrument, Publication-Data, Track, Audio, and Audio-Format. The "Album" type is currently selected, highlighted with a darker background.

# Functions

As you build your schema, you can view the JSON code by clicking on 'View JSON' .

This code can be:

- Downloaded to your local storage
- Copied to the clipboard
- Saved onto the file uploader for easy selection

Other features include:

- Checked for validation
- Reset to start over
- Scroll to top (appears when you scroll down on window)

```
1 {  
2   "info": {  
3     "title": "Music Library",  
4     "package": "http://fake-audio.org/music-lib",  
5     "version": "1.0",  
6     "description": "This information model defines a library of audio tracks, organized by album",  
7     "license": "CC0-1.0",  
8     "exports": ["Library", "Album", "Track"]  
9   },  
10  "types": [  
11    {"Library": "MapOf", "Barcode": "Album", "(1)", "", []},  
12    {"Barcode": "String", "Value": "A UPC-A barcode is 12 digits", []},  
13    {"Album": "Object", "Label": "The label for the album", []},  
14    {"Artist": "Artist", "Label": "artist associated with this album"},  
15    {"title": "String", "Label": "commonly known title for this album"},  
16    {"Publication-Data": "Object", "Label": "metadata about publication"},  
17    {"Record": "Object", "Label": "information about track description"},  
18    {"Cover-Art": "Image", "Label": "cover art image for this album"},  
19  ],  
20  "Records": [{"Record": "Object", "Label": "interesting information about the performers", [  
21    {"Artist": "Record", "Label": "who is this person"},  
22    {"Instruments": "ArrayOf", "Label": "what do they play"},  
23  ]},  
24  {"Instrument": "Enumerated", "Label": "collection of instruments (non-exhaustive)", [  
25    {"vocals": ""},  
26    {"guitar": ""},  
27    {"bass": ""},  
28    {"drums": ""},  
29    {"keyboards": ""},  
30    {"percussion": ""},  
31    {"brass": ""},  
32    {"woodwinds": ""},  
33    {"harmonica": ""}  
34  ]},  
35  {"Publication-Data": "Object", "Record": [], "Label": "who and when of publication", [  
36    {"label": "String", "Label": "name of record label"},  
37    {"rel_date": "String", "Label": "date"}, "when did they let this drop"  
38  ]},  
39  {"Track": "Record", "Label": "information about the individual audio tracks", [  
40    {"t_number": "Number", "Label": "track sequence number"}]
```

Click 'View Form' [View Form](#) to return to the Schema Creator.

The following slide shows you how to begin interacting  
with the page.

This applies to the following pages:

Schema Visualization

Schema Translation

Data Creation

Data Validation

Example Data Generation

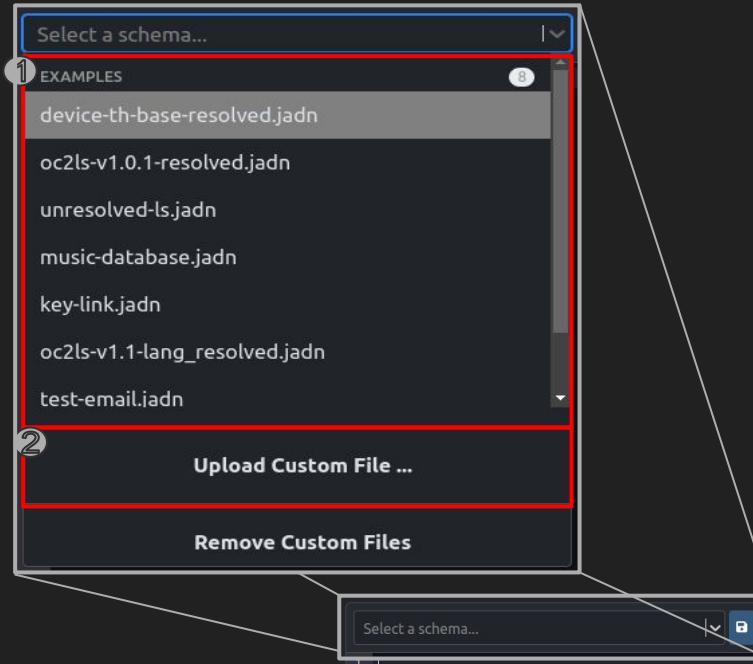
# Start with the File Loader :

Valid ✓

- ① Select a valid example schema  
or
- ② Upload a valid JADN Schema  
or
- ③ Type/ Paste in a valid JADN Schema in the code editor

You can also:

-  - format the Schema
-  - copy to clipboard
-  [Save the Schema to the file loader](#)



③

Valid ✘  

# Schema Visualization

View JADN schema in another language format:  
JIDL<sup>1</sup>, HTML, Markdown, PlantUML, GraphViz

1. [JADN Interface Definition Language \(IDL\)](#) is a textual representation of JADN type definitions

## Home

 <b>Creation</b> Create and edit JADN schemas using forms, view JADN schemas in JSON format.  Create schema compliant data instances (documents, messages).	 <b>Schema Visualization</b> Convert a JADN Schema into different visual representations.	 <b>Schema Translation</b> Translate a JADN Schema to another Schema format.  Translate a JSON Schema to a JADN Schema.
Schemas Data Instances	GraphViz HTML JIDL MarkDown PlantUML	JSON Relax (XML) XSD
 <b>Data Validation</b> Validate data instances in various data language formats against a JADN Schema.	 <b>Schema Transformation</b> Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).	 <b>Example Data Generation</b> Generate various example data instances based off of a Schema.
CBOR JSON Relax (XML)		

# Go to: Visualization

To Start : [click here](#)

Note: Only a JADN Schema can be uploaded.

After selecting a valid schema, select the desired languages you would like to convert the schema to.

Options on this page are for visual representation of JADN data. For Schema Translations (like to JSON or XML) [click here](#)

Visualize to...(select at least one) | 

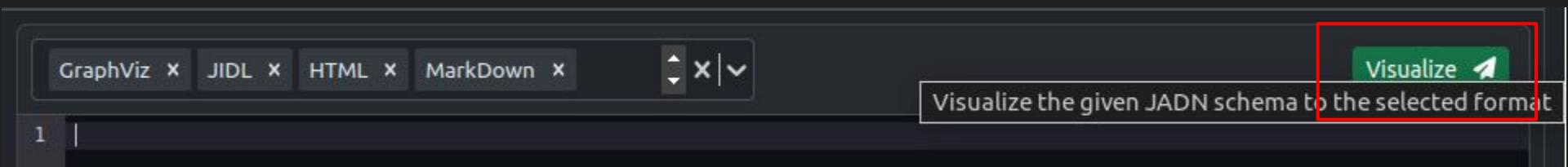


- GraphViz
- HTML
- JIDL
- MarkDown
- PlantUML

# Visualize

Once you have selected the languages, the green paper airplane button should light up allowing you to proceed to visualize the schema to the selected messages.

Click the Visualize button to proceed.



# Results: Selecting one language

The result will show you the code in the selected language.

The following features allow you to interact with the visualization as you need.



- Split view (currently shown) : To view the code and visualization simultaneously



- Pop out : See the visualization in a new tab window



- Download visualization as PDF



- Download visualization as an image



- Download code



- Copy code to Clipboard

HTML x ✖ | ↻

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>Music Library</title>
6     <style type="text/css"> /* Theme Colors - CC3322 as base using Adobe Kuler */
7   /* PDF Styles */
8   @page {
9     size: letter portrait;
10    @frame content_frame {
11      top: 16pt;
12      left: 16pt;
13    }
14  }
15  @media print {
16    #schema {
17      max-width: auto;
18    }
19  }
20  /* Standard Styles */
21  div>
```

**Schema**

```
package: "http://fake-audio.org/music-lib"
version: "1.0"
  title: "Music Library"
description: "This information model defines a library of audio tracks, organized by album"
  license: "CC0-1.0"
  exports: ["Library", "Album", "Track"]
  config: {}
```

**Compound Types**

**Library**

---

Library (MapOf(Barcode, Album)[1..\*])

**Album**

---

model for the album

# Results: Selecting multiple languages

You also have the ability to visualize multiple languages at once.

You can view the code for one or multiple languages at once by clicking on the header of each section.

Each language has its own applicable features as seen in each section.

The screenshot shows a user interface for viewing and interacting with code snippets across multiple languages. At the top, there are tabs for GraphViz, HTML, JIDL, and MarkDown, each with a close button. Below these tabs is a toolbar with icons for copy, paste, and other file operations. The main area contains five sections, each with its own tab and toolbar:

- GraphViz**: Shows a small preview of a graph visualization.
- HTML**: The active section, highlighted with a red border. It displays an HTML document structure with line numbers 1 through 21. The code includes doctype, html, head, meta, title, style, and media print rules.
- JIDL**: Shows a small preview of a JIDL code snippet.
- MarkDown**: Shows a small preview of a MarkDown code snippet.
- PlantUML**: Shows a small preview of a PlantUML code snippet.

Each section has its own toolbar below it, containing icons for copy, paste, and other file operations.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Music Library</title>
    <style type="text/css"> /* Theme Colors - CC3322 as base using Adobe Kuler */
    /* PDF Styles */
    @page {
      size: letter portrait;
      @frame content_frame {
        top: 16pt;
        left: 16pt;
      }
    }
    @media print {
      #schema {
        max-width: auto;
      }
    }
    /* Standard Styles */
  
```

# Schema Translation

Convert JADN Schema to JSON , Relax XML, or XSD.  
Convert JSON Schema to JADN.



## Home

**Creation**

Create and edit JADN schemas using forms, view JADN schemas in JSON format.

Create schema compliant data instances (documents, messages).

**Schema Visualization**

Convert a JADN Schema into different visual representations.

**Schema Translation**

Translate a JADN Schema to another Schema format.

Translate a JSON Schema to a JADN Schema.

Schemas Data Instances

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

**Data Validation**

Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Transformation**

Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).

**Example Data Generation**

Generate various example data instances based off of a Schema.

Go to: Translation

To Start : [click here](#)

Note: A JSON or JADN Schema can be uploaded.

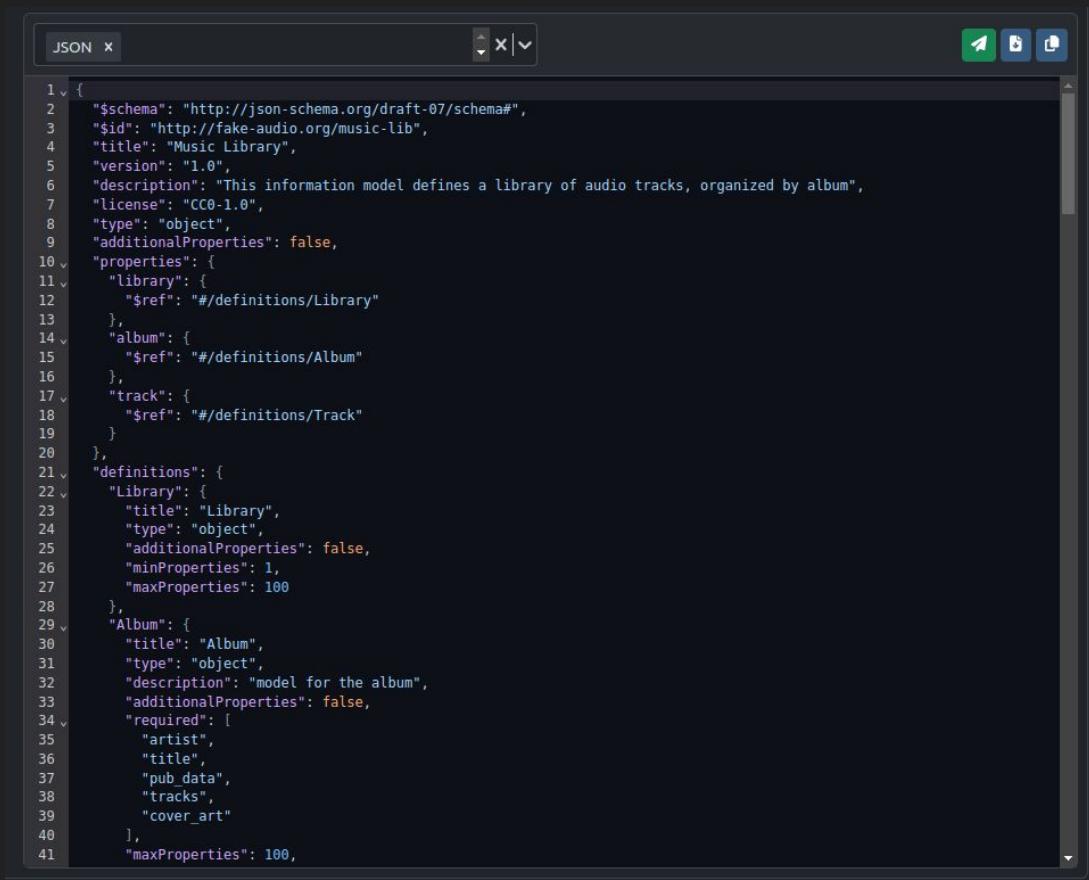
Similar to Schema Visualization,  
Schema Translation allows you to  
translate your schema into your  
desired language.

When selecting a schema, make sure  
to select the type of schema you want  
to translate from: JADN or JSON.



The screenshot shows the Schema Translation interface. On the left, there is a code editor window titled "music-database.jadn" containing JADN schema code. On the right, there is another code editor window titled "JSON x" containing JSON schema code. At the top center, there is a dropdown menu with the option "jadn" highlighted with a red box. Below the dropdown are several icons: a file icon, a "Valid" button, and other standard file operations.

```
music-database.jadn
1 v {
2 v   "info": {
3 v     "title": "Music Library",
4 v     "package": "http://fake-audio.org/music-lib".
1 v }
```



The screenshot shows the Schema Translation interface with the JSON schema displayed in the main window. The JSON code is identical to the one shown in the JADN editor above, indicating that the schema has been successfully translated from JADN to JSON.

```
JSON x
1 v {
2 v   "$schema": "http://json-schema.org/draft-07/schema#",
3 v   "$id": "http://fake-audio.org/music-lib",
4 v   "title": "Music Library",
5 v   "version": "1.0",
6 v   "description": "This information model defines a library of audio tracks, organized by album",
7 v   "license": "CC0-1.0",
8 v   "type": "object",
9 v   "additionalProperties": false,
10 v   "properties": {
11 v     "library": {
12 v       "$ref": "#/definitions/Library"
13 v     },
14 v     "album": {
15 v       "$ref": "#/definitions/Album"
16 v     },
17 v     "track": {
18 v       "$ref": "#/definitions/Track"
19 v     }
20 v   },
21 v   "definitions": {
22 v     "Library": {
23 v       "title": "Library",
24 v       "type": "object",
25 v       "additionalProperties": false,
26 v       "minProperties": 1,
27 v       "maxProperties": 100
28 v     },
29 v     "Album": {
30 v       "title": "Album",
31 v       "type": "object",
32 v       "description": "model for the album",
33 v       "additionalProperties": false,
34 v       "required": [
35 v         "artist",
36 v         "title",
37 v         "pub_data",
38 v         "tracks",
39 v         "cover_art"
40 v       ],
41 v       "maxProperties": 100,
42 v     }
43 v   }
44 v }
```

Result: Translated Schema to one language

# Result: Translated Schema to multiple languages

You can translate from JADN to JSON and XML (XSD or Relax) or from JSON to JADN.

Schema Translation

music-database.jadn      X | V      Jadn      X | V      Valid     

```
1 v {  
2 v   "info": [  
3 v     "title": "Music Library",  
4 v     "package": "http://fake-audio.org/music-lib",  
5 v     "version": "1.0",  
6 v     "description": "This information model defines a library of audio tracks, organized by album",  
7 v     "license": "CC0-1.0",  
8 v     "exports": ["Library", "Album", "Track"]  
9 },  
10 v  
11 v   "types": [  
12 v     ["Library", "MapOf", ["+Barcode", "+Album", "{1}", "", []],  
13 v     ["Barcode", "String", ["%\\d{12}"], "A UPC-A barcode is 12 digits", []],  
14 v     ["Album", "Record", [], "model for the album", [  
15 v       [1, "artist", "Artist", [], "artist associated with this album"],  
16 v       [2, "title", "String", [], "commonly known title for this album"],  
17 v       [3, "pub_data", "Publication-Data", [], "metadata about album publication"],  
18 v       [4, "tracks", "ArrayOf", ["*Track", "0"], "individual track descriptions"],  
19 v       [5, "cover_art", "Cover-Art", [], "cover art image for this album"]  
20 v     ]],  
21 v     ["Artist", "Record", [], "interesting information about the performers", [  
22 v       [1, "artist_name", "String", [], "who is this person"],  
23 v       [2, "instruments", "ArrayOf", ["*Instrument", "0"], "and what do they play"]  
24 v     ]],  
25 v     ["Instrument", "Enumerated", [], "collection of instruments (non-exhaustive)", [  
26 v       [1, "vocals", ""],  
27 v       [2, "guitar", ""],  
28 v       [3, "bass", ""],  
29 v       [4, "drums", ""],  
30 v       [5, "keyboards", ""],  
31 v       [6, "percussion", ""],  
32 v       [7, "brass", ""],  
33 v       [8, "woodwinds", ""],  
34 v       [9, "harmonica", ""]  
35 v     ]],  
36 v     ["Publication-Data", "Record", [], "who and when of publication", [  
37 v       [1, "label", "String", [], "name of record label"],  
38 v       [2, "rel_date", "String", ["*date"], "and when did they let this drop"]  
39 v     ]],  
40 v     ["Track", "Record", [], "information about the individual audio tracks", [  
41 v       [1, "t_number", "Number", [], "track sequence number"],  
42 v       [2, "title", "String", [], "track title"],  
43 v     ]]  
44 v   ]]  
45 v }
```

XSD      JSON

XSD

```
1 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:jadn="jadn_base_types.xsd" namespace="jadn_base_types" />  
2   <xsd:import schemaLocation="../../data/xsd/jadn_base_types.xsd" />  
3   <xsd:complexType name="Library">  
4     <xsd:sequence>  
5       <xsd:element id="library_map" name="library_map" minOccurs="1" maxOccurs="unbounded">  
6         <xsd:complexType>  
7           <xsd:sequence>  
8             <xsd:element id="library_map_barcode" name="Barcode" />  
9             <xsd:element id="library_map_album" name="Album" />  
10            <xsd:sequence>  
11              <xsd:complexType>  
12                <xsd:element>  
13                  <xsd:sequence>  
14                    <xsd:element>  
15                      <xsd:simpleType name="Barcode">  
16                        <xsd:annotation>  
17                          <xsd:documentation>A UPC-A barcode is 12 digits</xsd:documentation>  
18                        <xsd:annotation>  
19                          <xsd:restriction base="jadn:String">  
20                            <xsd:pattern value="\d{12}" />  
21                          </xsd:restriction>  
22                      </xsd:annotation>  
23                  </xsd:sequence>  
24                </xsd:complexType>  
25            </xsd:sequence>  
26          </xsd:element>  
27        </xsd:sequence>  
28      </xsd:complexType>  
29    <xsd:annotation>  
30      <xsd:documentation>A UPC-A barcode is 12 digits</xsd:documentation>  
31    </xsd:annotation>  
32  <xsd:restriction base="jadn:String">  
33    <xsd:pattern value="\d{12}" />  
34  </xsd:restriction>
```

JSON

```
1 v {  
2   "$schema": "http://json-schema.org/draft-07/schema#",  
3   "$id": "http://fake-audio.org/music-lib",  
4   "title": "Music Library",  
5   "version": "1.0",  
6   "description": "This information model defines a library of audio tracks, organized by album",  
7   "license": "CC0-1.0",  
8   "type": "object",  
9   "additionalProperties": false,  
10  "properties": {  
11    "library": {  
12      "$ref": "#/definitions/Library"
```

# Data Creation

Create valid data instances from a JADN Schema

JADN Sandbox Home **Creation** ▾ Visualization Translation Validation Transformation Generation About

Home

**Creation**

Create and edit JADN schemas using forms, view JADN schemas in JSON format.

Create schema compliant data instances (documents, messages).

**Schema Creation**

**Data Creation**

**Schema Visualization**

Convert a JADN Schema into different visual representations.

**Schema Translation**

Translate a JADN Schema to another Schema format.

Translate a JSON Schema to a JADN Schema.

Schemas Data Instances

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

**Data Validation**

Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Transformation**

Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).

**Example Data Generation**

Generate various example data instances based off of a Schema.

localhost:8082/create/message v0.10.0\_1705587211962

Go to: Creation  
When the drop down menu appears, select Data Creation

To Start : [click here](#)

Note: Only a JADN Schema can be uploaded.

After selecting a schema, you should be able to select a data type.

Note: This drop down menu is based on the Schema exports.

Once you have selected a data type, a form will appear to help guide you in building data that is valid against the selected schema.

The screenshot shows a schema editor interface with the following elements:

- Top Bar:** A search bar containing "Library", a lock icon, and three icons for "View JSON", "Upload", and "Download".
- Left Sidebar:** A dropdown menu with options: "Library" (selected), "Album", and "Track".
- Form Fields:** The "Library" section is expanded, showing:
  - barcode\***: A field with placeholder text "A UPC-A barcode is 12 digits" and a regular expression input field containing "\d{12}".
  - album\***: A field with placeholder text "model for the album".
  - artist\***: A field with placeholder text "artist associated with this album".
  - artist\_name\***: A field with placeholder text "who is this person".

# View JSON / Creator

After filling out the form, you can click 'View JSON' to see the data you have created in JSON.

[View JSON](#)

Click 'View Creator' to go back to the form.

[View Creator](#)

The JSON can also be:



- Downloaded to your local storage
- Copied to the clipboard

A screenshot of a JSON editor window titled "Library". The window contains the following JSON code:

```
1 v {
2 v   "Library": {
3 v     "123456789123": {
4 v       "artist": {
5 v         "artist_name": "the Beatles",
6 v         "instruments": ["guitar", "guitar"]
7 v       },
8 v       "title": "Hey Jude"
9 v     }
10 v   }
11 }
```

The "View Creator" button in the top right corner is highlighted with a red box.

# Data Validation

Validate data instances in JSON, XML, or CBOR against a  
JADN Schema

## Home

**Creation**

Create and edit JADN schemas using forms, view JADN schemas in JSON format.

Create schema compliant data instances (documents, messages).

**Schema Visualization**

Convert a JADN Schema into different visual representations.

**Schema Translation**

Translate a JADN Schema to another Schema format.

Translate a JSON Schema to a JADN Schema.

Schemas Data Instances

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

**Data Validation**

Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Transformation**

Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).

**Example Data Generation**

Generate various example data instances based off of a Schema.

# Go to: Validation

To Start : [click here](#)

Note: Only a JADN Schema can be uploaded.

① Select data to validate.

② Make sure the correct data format is selected.

③ Make sure the correct data type being validated is selected.

In this example, the data is a json file being validated against the Library type from the music-database.jadn

The screenshot shows a user interface for validating JADN data. At the top, there are three dropdown menus labeled 1, 2, and 3, each with a red box around it. Menu 1 contains 'query\_pairs\_a.json'. Menu 2 contains 'json'. Menu 3 contains 'Library'. To the right of these menus is a green 'Validate' button with a checkmark icon, and below it are three small blue icons. Below the menus is a code editor window displaying the following JSON code:

```
1 {  
2   "action": "query",  
3   "target": {  
4     "features": [  
5       "pairs"  
6     ]  
7   }  
8 }
```

## Select data to validate

Starting by uploading data:

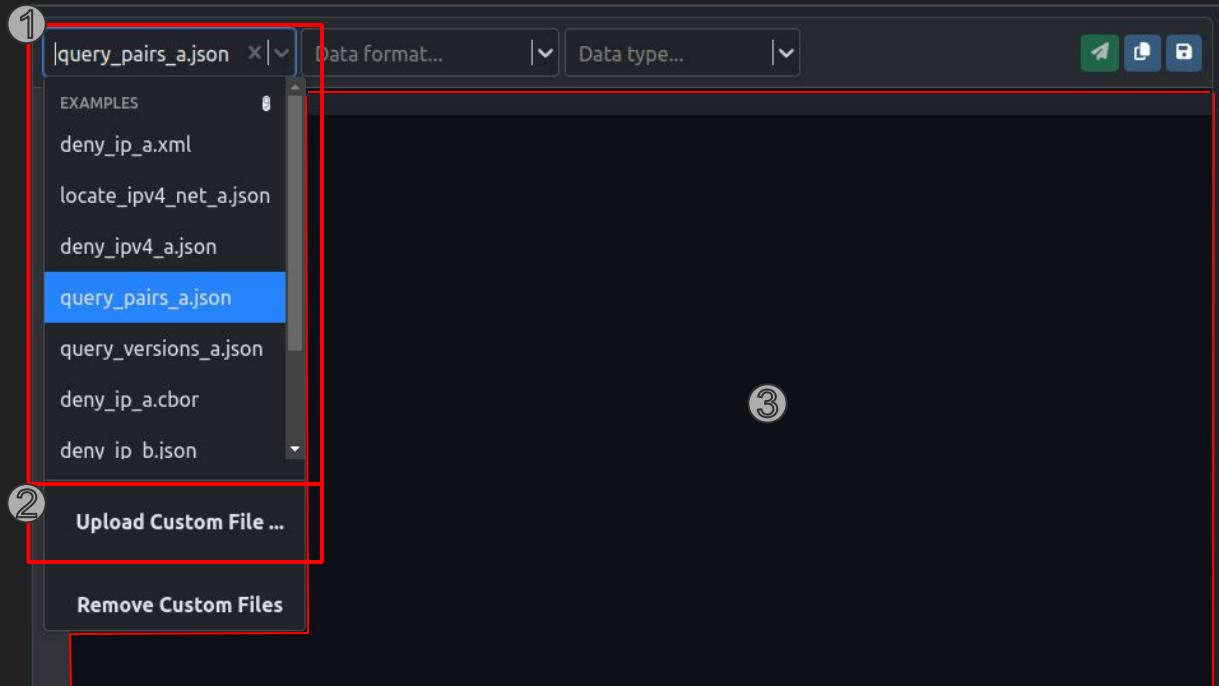
- ① Select example data

or

- ② Upload a custom file

or

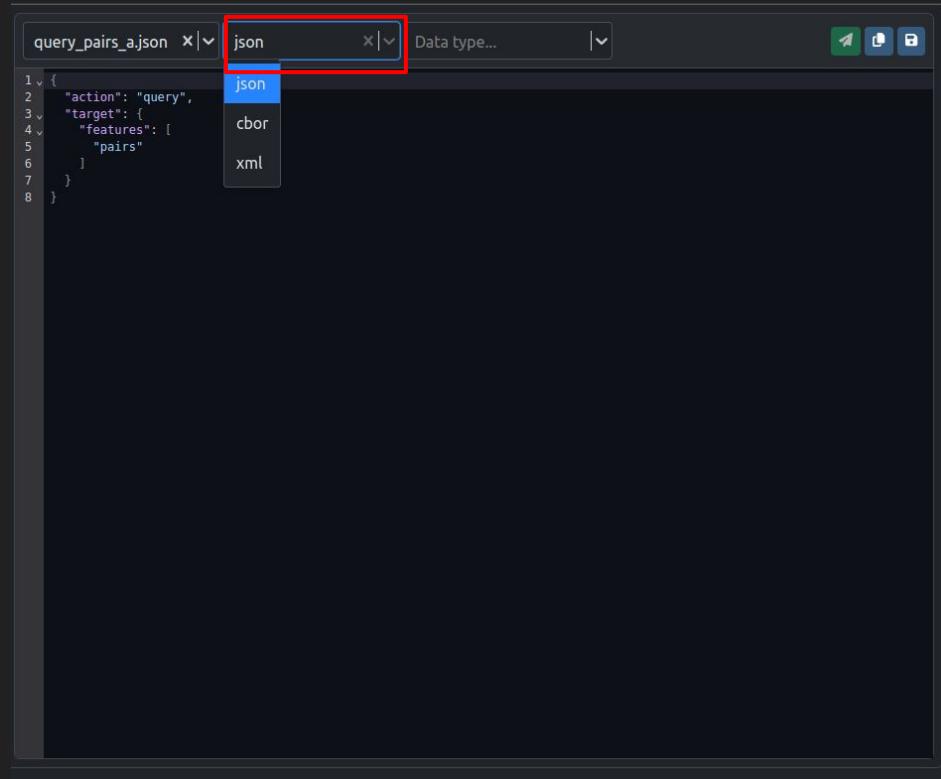
- ③ type/paste directly into code editor.



## Make sure to select the correct data format

The JADN Sandbox can currently validate JSON, CBOR, and XML data against a JADN Schema.

If using a data file, the data format drop down may be pre-selected.

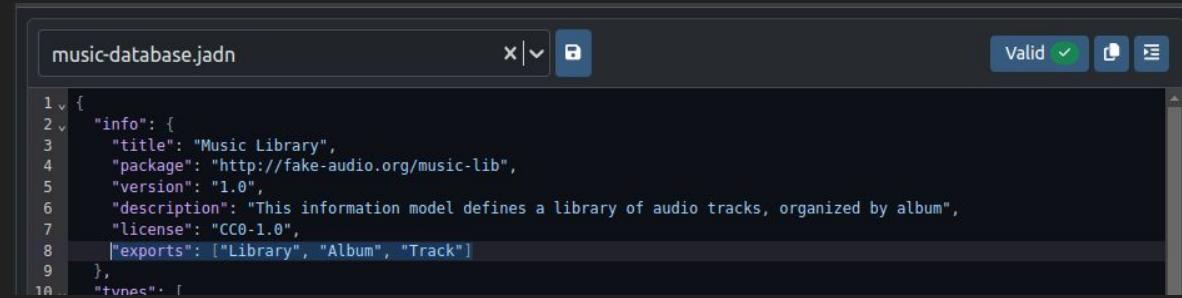


## Make sure the correct data type is selected

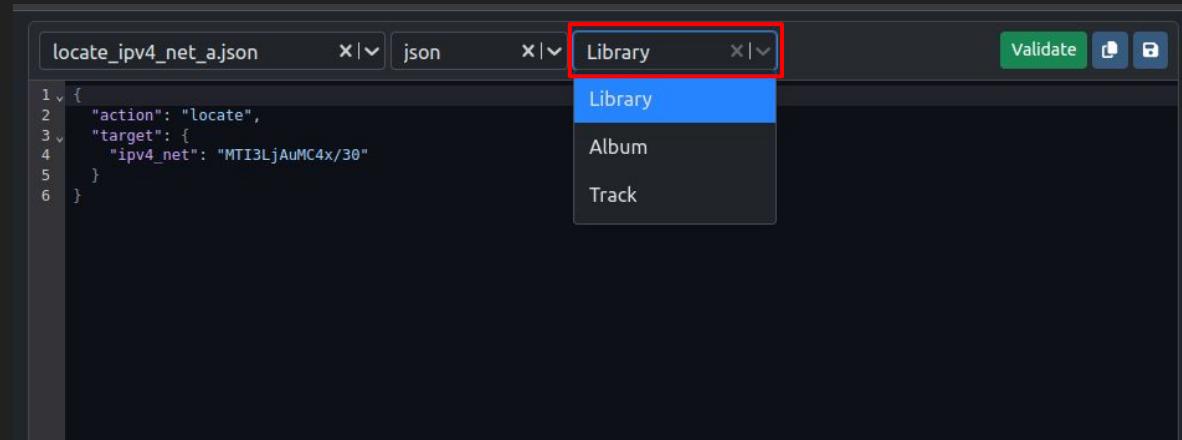
The Data types available to validate against are based off of the exports from the JADN Schema file.

In this example, notice that the uploaded schema music-database.jadn has exports that correspond to the valid datatypes:

Library, Album, Track



```
music-database.jadn
1 v {
2 v   "info": {
3 v     "title": "Music Library",
4 v     "package": "http://fake-audio.org/music-lib",
5 v     "version": "1.0",
6 v     "description": "This information model defines a library of audio tracks, organized by album",
7 v     "license": "CC0-1.0",
8 v     "exports": ["Library", "Album", "Track"]
9 v   },
10 v }
```



```
locate_ipv4_net_a.json
1 v {
2 v   "action": "locate",
3 v   "target": {
4 v     "ipv4_net": "MTI3LjAuMC4x/30"
5 v   }
6 v }
```

Library

Album

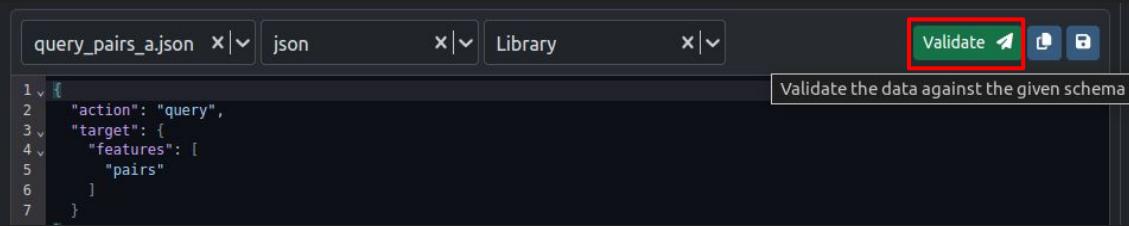
Track

# Validate

Once you have selected:

- A Schema,
- Data,
- Data format, and
- Data type

Click the bright, green “Validate” button.



The screenshot shows a user interface for validating JSON data. At the top, there are three tabs: "query\_pairs\_a.json" (selected), "json" (disabled), and "Library" (disabled). Below the tabs is a code editor containing the following JSON code:

```
1 v [
2   "action": "query",
3   "target": {
4     "features": [
5       "pairs"
6     ]
7   }
]
```

To the right of the code editor is a toolbar with several icons. One icon, labeled "Validate" with a green checkmark, is highlighted with a red rectangular border. A tooltip below the toolbar reads "Validate the data against the given schema".

# Invalid Results

An invalid data file will return errors to help fix the data.

The screenshot shows a code editor window with a dark theme. The title bar includes tabs for 'query\_pairs\_a.json', 'json', and 'Library'. On the right side of the title bar are icons for saving, opening, and closing. The main area displays the following JSON code:

```
1 {  
2   "action": "query",  
3   "target": {  
4     "features": [  
5       "pairs"  
6     ]  
7   }  
8 }
```

Below the code, a series of five red error notifications are listed, each with an exclamation icon and a message:

- field required at artist
- object of type  
'builtin\_function\_or\_method' has  
no len() at title
- field required at pub\_data
- field required at tracks
- field required at cover\_art

A 'Clear All' button is located at the bottom right of the error list.

# Valid Results

A valid data file will return a green success notification.

The screenshot shows a JSON validation interface with the following details:

- File type: json
- Validation status: Validation success (indicated by a green checkmark icon)
- Content: A valid JSON object representing an album. The object includes fields for artist, title, publication data, tracks, and cover art.

```
1 v {
2 v   "artist": {
3 v     "artist_name": "the Beatles",
4 v     "instruments": ["vocals"]
5 v   },
6 v   "title": "Hey Jude",
7 v   "pub_data": {
8 v     "label": "Capital Records",
9 v     "rel_date": "1979-05-11"
10 v   },
11 v   "tracks": [
12 v     {
13 v       "t_number": 1,
14 v       "title": "Can't Buy Me Love",
15 v       "length": "00:02:19",
16 v       "featured": [],
17 v       "audio": {
18 v         "a_format": "MP3",
19 v         "a_content": "dGhlIEJlYXRsZXM="
20 v       }
21     }
22   ],
23 v   "cover_art": {
24 v     "i_format": "JPG",
25 v     "i_content": "dGhlIEJlYXRsZXM="
26 v   }
27 }
```

# Schema Transformation

Strip comments from Schemas or Attempt to resolve an unresolved schema



## Home

**Creation**

Create and edit JADN schemas using forms, view JADN schemas in JSON format.

Create schema compliant data instances (documents, messages).

**Schema Visualization**

Convert a JADN Schema into different visual representations.

**Schema Translation**

Translate a JADN Schema to another Schema format.

Translate a JSON Schema to a JADN Schema.

Schemas Data Instances

GraphViz HTML JIDL MarkDown PlantUML

JSON Relax (XML) XSD

**Data Validation**

Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Transformation**

Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).

**Example Data Generation**

Generate various example data instances based off of a Schema.

Theme ▾

v0.10.0\_1705587211962

# Go to: Transformation

To Start :

Select at least one schema to upload.

Notice that there is no code editor. Therefore, schemas must be uploaded.

Here you can see 2 schemas selected to start:

- unresolved-ls.jadn
- oc2ls-v1.1-lang\_resolved.jadn

The screenshot shows a schema editor interface with two tabs open:

- unresolved-ls.jadn**: This tab displays the JSON content of the 'unresolved-ls.jadn' schema. It includes sections for info, package, version, title, description, namespaces, exports, and types. The types section defines several record structures, such as 'Artifact', 'Command-ID', and 'Device'.
- oc2ls-v1.1-lang\_resolved.jadn**: This tab displays the JSON content of the 'oc2ls-v1.1-lang\_resolved.jadn' schema. It includes sections for info, package, title, description, exports, and types. The types section defines a 'Record' type for 'Command' with fields 'action', 'target', and 'args'.

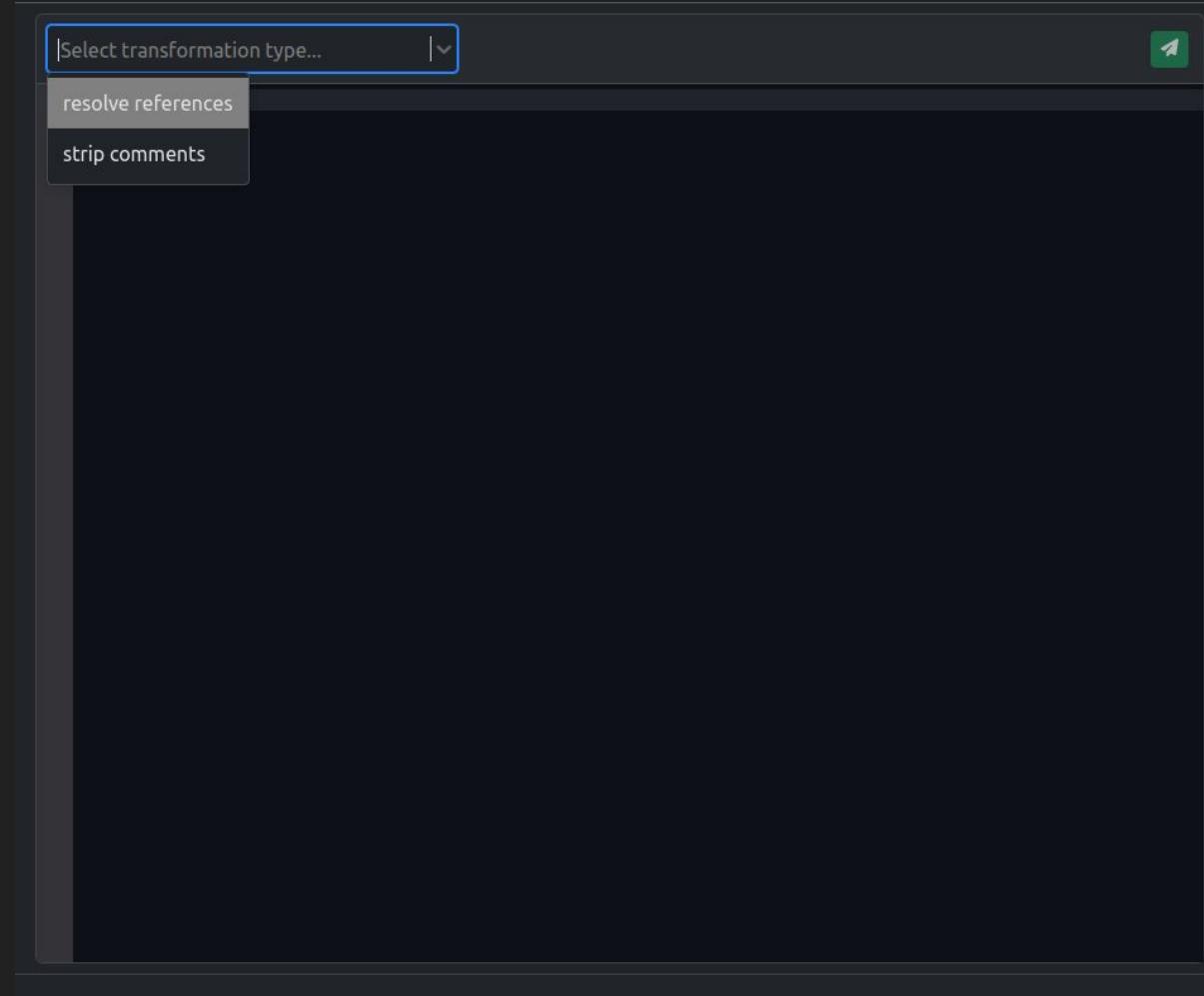
After selecting the desired schemas, you can select your desired transformation:

### Resolve references

- Create a single JADN schema from an unresolved schema and its imports

### Strip comments

- Remove comments from a JADN schema to reduce size or visual clutter



# Resolve References <sup>1</sup>

For resolving references, you will need to select a base file. This file is the file you would like to be resolved.

The other uploaded files will be used to help resolve the chosen base file.

In this example, the base file is :  
unresolved-ls.jadn

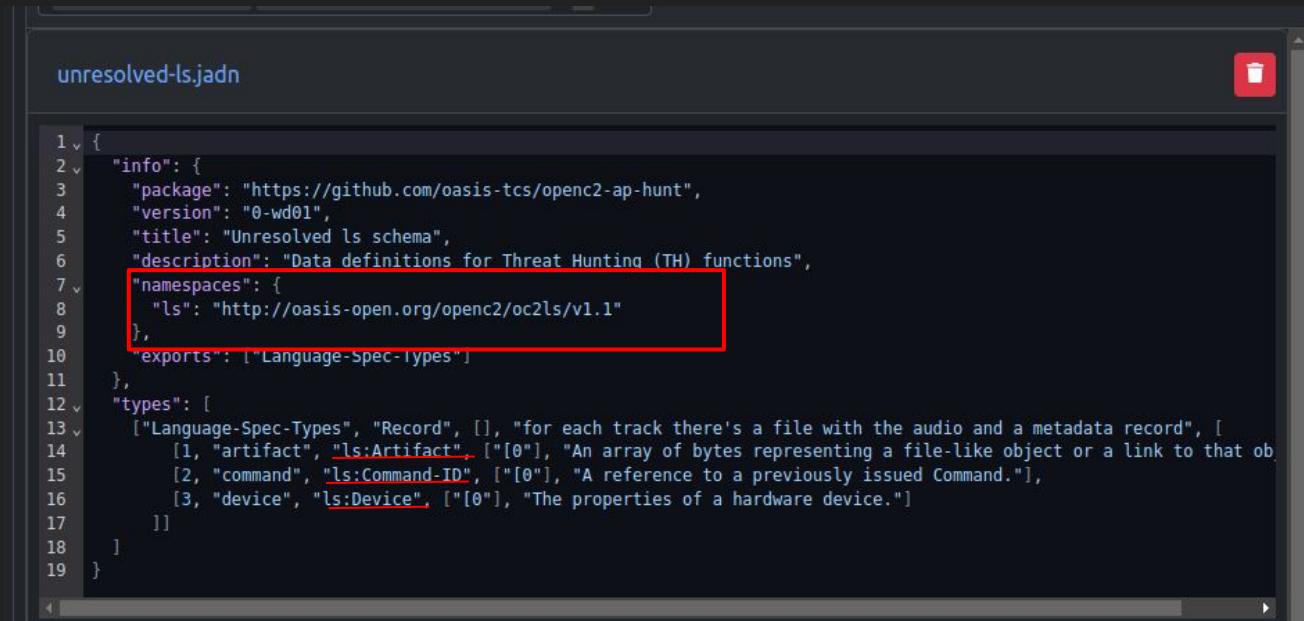
1. A resolved Schema is a schema without references.

The screenshot shows the OpenSCAP interface with the 'Resolve References' dialog open. The dialog has a red border and contains the text 'resolve references' and 'x | v'. To the right of the dialog is a dropdown menu labeled 'Select base file...' with two options: 'unresolved-ls.jadn' and 'oc2ls-v1.1-lang\_resolved.jadn'. Below the dialog is a code editor window titled 'unresolved-ls.jadn' containing the following JSON-like schema definition:

```
1 v {  
2 v   "info": {  
3 v     "package": "https://github.com/oasis-tcs/openc2-ap-hunt",  
4 v     "version": "0-wd01",  
5 v     "title": "Unresolved ls schema",  
6 v     "description": "Data definitions for Threat Hunting (TH) functions",  
7 v     "namespaces": {  
8 v       "ls": "http://oasis-open.org/openc2/oc2ls/v1.1"  
9 v     },  
10 v    "exports": ["Language-Spec-Types"]  
11 v  },  
12 v  "types": [  
13 v    ["Language-Spec-Types", "Record", [], "for each track there's a file with the audio and a metadata record", [  
14 v      [1, "artifact", "ls:Artifact", "[0]", "An array of bytes representing a file-like object or a link to that ob  
15 v      [2, "command", "ls:Command-ID", "[0]", "A reference to a previously issued Command."],  
16 v      [3, "device", "ls:Device", "[0]", "The properties of a hardware device."]  
17 v    ]]  
18 v  ]  
19 v}
```

# What is an unresolved Schema?

A Schema is unresolved if it has namespaces (references to types defined in other schemas).



```
unresolved-ls.jadn
1 v {
2 v   "info": {
3 v     "package": "https://github.com/oasis-tcs/openc2-ap-hunt",
4 v     "version": "0-wd01",
5 v     "title": "Unresolved ls schema",
6 v     "description": "Data definitions for Threat Hunting (TH) functions",
7 v     "namespaces": {
8 v       "ls": "http://oasis-open.org/openc2/oc2ls/v1.1"
9 v     },
10 v     "exports": ["Language-Spec-Types"]
11 },
12 v   "types": [
13 v     ["Language-Spec-Types", "Record", [], "for each track there's a file with the audio and a metadata record", [
14 v       [1, "artifact", "ls:Artifact", "[[0]", "An array of bytes representing a file-like object or a link to that ob
15 v       [2, "command", "ls:Command-ID", "[[0]", "A reference to a previously issued Command."]
16 v       [3, "device", "ls:Device", "[[0]", "The properties of a hardware device."]
17 v     ]]
18   ]
19 }
```

Click Transform to resolve the base file.



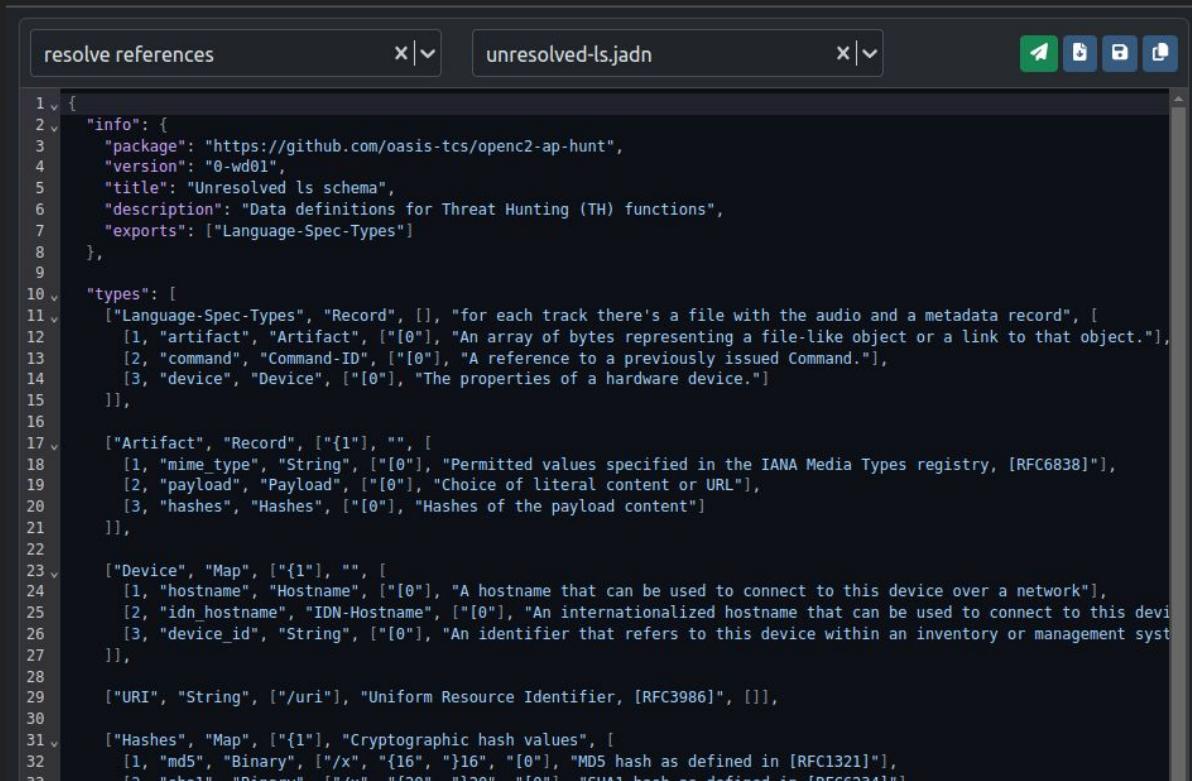
# Result: A resolved Schema

Notice there is no namespace and that the types from the unresolved schema can be referenced within the resolved schema.

Ls: Artifact → Artifact

Ls: Command-ID → Command-ID

Ls: Device → Device



```
1 v {
2 v   "info": {
3 v     "package": "https://github.com/oasis-tcs/openc2-ap-hunt",
4 v     "version": "0-wd01",
5 v     "title": "Unresolved ls schema",
6 v     "description": "Data definitions for Threat Hunting (TH) functions",
7 v     "exports": ["Language-Spec-Types"]
8 },
9
10 v   "types": [
11 v     ["Language-Spec-Types", "Record", [], "for each track there's a file with the audio and a metadata record", [
12 v       [1, "artifact", "Artifact", "[[0]]", "An array of bytes representing a file-like object or a link to that object."],
13 v       [2, "command", "Command-ID", "[[0]]", "A reference to a previously issued Command."],
14 v       [3, "device", "Device", "[[0]]", "The properties of a hardware device."]
15 ],
16
17 v     ["Artifact", "Record", [{"1": "", [
18 v       [1, "mime_type", "String", "[[0]]", "Permitted values specified in the IANA Media Types registry, [RFC6838]"],
19 v       [2, "payload", "Payload", "[[0]]", "Choice of literal content or URL"],
20 v       [3, "hashes", "Hashes", "[[0]]", "Hashes of the payload content"]
21 ],
22
23 v       ["Device", "Map", [{"1": "", [
24 v         [1, "hostname", "Hostname", "[[0]]", "A hostname that can be used to connect to this device over a network"],
25 v         [2, "idn_hostname", "IDN-Hostname", "[[0]]", "An internationalized hostname that can be used to connect to this devi
26 v         [3, "device_id", "String", "[[0]]", "An identifier that refers to this device within an inventory or management syst
27 ]],
28
29 v       ["URI", "String", ["/uri"], "Uniform Resource Identifier, [RFC3986]", []],
30
31 v       ["Hashes", "Map", [{"1": "", "Cryptographic hash values", [
32 v         [1, "md5", "Binary", [/X/, {"16": "[0]"}, "MD5 hash as defined in [RFC1321]"],
33 v         [2, "sha1", "Binary", [/x/, {"16": "[0]"}, "SHA1 hash as defined in [RFC6234]"]
34
35 v       ]],
36
37 v     ]],
38
39 v   ]],
40
41 v   ["File", "Record", [{"1": "", [
42 v     [1, "path", "String", "[[0]]", "The path to the file being listed."],
43 v     [2, "size", "Number", "[[0]]", "The size of the file in bytes."]
44 ],
45
46 v   ]],
47
48 v ]],
49
50 v ]],
51
52 v ]],
53
54 v ]],
55
56 v ]],
57
58 v ]],
59
60 v ]],
61
62 v ]],
63
64 v ]],
65
66 v ]],
67
68 v ]],
69
70 v ]],
71
72 v ]],
73
74 v ]],
75
76 v ]],
77
78 v ]],
79
80 v ]],
81
82 v ]],
83
84 v ]],
85
86 v ]],
87
88 v ]],
89
90 v ]],
91
92 v ]],
93
94 v ]],
95
96 v ]],
97
98 v ]],
99
100 v ]],
101
102 v ]],
103
104 v ]],
105
106 v ]],
107
108 v ]],
109
110 v ]],
111
112 v ]],
113
114 v ]],
115
116 v ]],
117
118 v ]],
119
120 v ]],
121
122 v ]],
123
124 v ]],
125
126 v ]],
127
128 v ]],
129
129 v ]],
130
131 v ]],
132
132 v ]],
133
133 v ]],
134
134 v ]],
135
135 v ]],
136
136 v ]],
137
137 v ]],
138
138 v ]],
139
139 v ]],
140
140 v ]],
141
141 v ]],
142
142 v ]],
143
143 v ]],
144
144 v ]],
145
145 v ]],
146
146 v ]],
147
147 v ]],
148
148 v ]],
149
149 v ]],
150
150 v ]],
151
151 v ]],
152
152 v ]],
153
153 v ]],
154
154 v ]],
155
155 v ]],
156
156 v ]],
157
157 v ]],
158
158 v ]],
159
159 v ]],
160
160 v ]],
161
161 v ]],
162
162 v ]],
163
163 v ]],
164
164 v ]],
165
165 v ]],
166
166 v ]],
167
167 v ]],
168
168 v ]],
169
169 v ]],
170
170 v ]],
171
171 v ]],
172
172 v ]],
173
173 v ]],
174
174 v ]],
175
175 v ]],
176
176 v ]],
177
177 v ]],
178
178 v ]],
179
179 v ]],
180
180 v ]],
181
181 v ]],
182
182 v ]],
183
183 v ]],
184
184 v ]],
185
185 v ]],
186
186 v ]],
187
187 v ]],
188
188 v ]],
189
189 v ]],
190
190 v ]],
191
191 v ]],
192
192 v ]],
193
193 v ]],
194
194 v ]],
195
195 v ]],
196
196 v ]],
197
197 v ]],
198
198 v ]],
199
199 v ]],
200
200 v ]],
201
201 v ]],
202
202 v ]],
203
203 v ]],
204
204 v ]],
205
205 v ]],
206
206 v ]],
207
207 v ]],
208
208 v ]],
209
209 v ]],
210
210 v ]],
211
211 v ]],
212
212 v ]],
213
213 v ]],
214
214 v ]],
215
215 v ]],
216
216 v ]],
217
217 v ]],
218
218 v ]],
219
219 v ]],
220
220 v ]],
221
221 v ]],
222
222 v ]],
223
223 v ]],
224
224 v ]],
225
225 v ]],
226
226 v ]],
227
227 v ]],
228
228 v ]],
229
229 v ]],
230
230 v ]],
231
231 v ]],
232
232 v ]],
233
233 v ]],
234
234 v ]],
235
235 v ]],
236
236 v ]],
237
237 v ]],
238
238 v ]],
239
239 v ]],
240
240 v ]],
241
241 v ]],
242
242 v ]],
243
243 v ]],
244
244 v ]],
245
245 v ]],
246
246 v ]],
247
247 v ]],
248
248 v ]],
249
249 v ]],
250
250 v ]],
251
251 v ]],
252
252 v ]],
253
253 v ]],
254
254 v ]],
255
255 v ]],
256
256 v ]],
257
257 v ]],
258
258 v ]],
259
259 v ]],
260
260 v ]],
261
261 v ]],
262
262 v ]],
263
263 v ]],
264
264 v ]],
265
265 v ]],
266
266 v ]],
267
267 v ]],
268
268 v ]],
269
269 v ]],
270
270 v ]],
271
271 v ]],
272
272 v ]],
273
273 v ]],
274
274 v ]],
275
275 v ]],
276
276 v ]],
277
277 v ]],
278
278 v ]],
279
279 v ]],
280
280 v ]],
281
281 v ]],
282
282 v ]],
283
283 v ]],
284
284 v ]],
285
285 v ]],
286
286 v ]],
287
287 v ]],
288
288 v ]],
289
289 v ]],
290
290 v ]],
291
291 v ]],
292
292 v ]],
293
293 v ]],
294
294 v ]],
295
295 v ]],
296
296 v ]],
297
297 v ]],
298
298 v ]],
299
299 v ]],
300
300 v ]],
301
301 v ]],
302
302 v ]],
303
303 v ]],
304
304 v ]],
305
305 v ]],
306
306 v ]],
307
307 v ]],
308
308 v ]],
309
309 v ]],
310
310 v ]],
311
311 v ]],
312
312 v ]],
313
313 v ]],
314
314 v ]],
315
315 v ]],
316
316 v ]],
317
317 v ]],
318
318 v ]],
319
319 v ]],
320
320 v ]],
321
321 v ]],
322
322 v ]],
323
323 v ]],
324
324 v ]],
325
325 v ]],
326
326 v ]],
327
327 v ]],
328
328 v ]],
329
329 v ]],
330
330 v ]],
331
331 v ]],
332
332 v ]],
333
333 v ]],
334
334 v ]],
335
335 v ]],
336
336 v ]],
337
337 v ]],
338
338 v ]],
339
339 v ]],
340
340 v ]],
341
341 v ]],
342
342 v ]],
343
343 v ]],
344
344 v ]],
345
345 v ]],
346
346 v ]],
347
347 v ]],
348
348 v ]],
349
349 v ]],
350
350 v ]],
351
351 v ]],
352
352 v ]],
353
353 v ]],
354
354 v ]],
355
355 v ]],
356
356 v ]],
357
357 v ]],
358
358 v ]],
359
359 v ]],
360
360 v ]],
361
361 v ]],
362
362 v ]],
363
363 v ]],
364
364 v ]],
365
365 v ]],
366
366 v ]],
367
367 v ]],
368
368 v ]],
369
369 v ]],
370
370 v ]],
371
371 v ]],
372
372 v ]],
373
373 v ]],
374
374 v ]],
375
375 v ]],
376
376 v ]],
377
377 v ]],
378
378 v ]],
379
379 v ]],
380
380 v ]],
381
381 v ]],
382
382 v ]],
383
383 v ]],
384
384 v ]],
385
385 v ]],
386
386 v ]],
387
387 v ]],
388
388 v ]],
389
389 v ]],
390
390 v ]],
391
391 v ]],
392
392 v ]],
393
393 v ]],
394
394 v ]],
395
395 v ]],
396
396 v ]],
397
397 v ]],
398
398 v ]],
399
399 v ]],
400
400 v ]],
401
401 v ]],
402
402 v ]],
403
403 v ]],
404
404 v ]],
405
405 v ]],
406
406 v ]],
407
407 v ]],
408
408 v ]],
409
409 v ]],
410
410 v ]],
411
411 v ]],
412
412 v ]],
413
413 v ]],
414
414 v ]],
415
415 v ]],
416
416 v ]],
417
417 v ]],
418
418 v ]],
419
419 v ]],
420
420 v ]],
421
421 v ]],
422
422 v ]],
423
423 v ]],
424
424 v ]],
425
425 v ]],
426
426 v ]],
427
427 v ]],
428
428 v ]],
429
429 v ]],
430
430 v ]],
431
431 v ]],
432
432 v ]],
433
433 v ]],
434
434 v ]],
435
435 v ]],
436
436 v ]],
437
437 v ]],
438
438 v ]],
439
439 v ]],
440
440 v ]],
441
441 v ]],
442
442 v ]],
443
443 v ]],
444
444 v ]],
445
445 v ]],
446
446 v ]],
447
447 v ]],
448
448 v ]],
449
449 v ]],
450
450 v ]],
451
451 v ]],
452
452 v ]],
453
453 v ]],
454
454 v ]],
455
455 v ]],
456
456 v ]],
457
457 v ]],
458
458 v ]],
459
459 v ]],
460
460 v ]],
461
461 v ]],
462
462 v ]],
463
463 v ]],
464
464 v ]],
465
465 v ]],
466
466 v ]],
467
467 v ]],
468
468 v ]],
469
469 v ]],
470
470 v ]],
471
471 v ]],
472
472 v ]],
473
473 v ]],
474
474 v ]],
475
475 v ]],
476
476 v ]],
477
477 v ]],
478
478 v ]],
479
479 v ]],
480
480 v ]],
481
481 v ]],
482
482 v ]],
483
483 v ]],
484
484 v ]],
485
485 v ]],
486
486 v ]],
487
487 v ]],
488
488 v ]],
489
489 v ]],
490
490 v ]],
491
491 v ]],
492
492 v ]],
493
493 v ]],
494
494 v ]],
495
495 v ]],
496
496 v ]],
497
497 v ]],
498
498 v ]],
499
499 v ]],
500
500 v ]]
```

# Result: Strip Comments

When stripping comments, results will return all schemas without comments.

The screenshot shows a code editor interface with two tabs. The top bar has a search field containing "strip comments" with a red box highlighting it. The tabs are labeled "unresolved-ls" and "oc2ls-v1.1-lang\_resolved".

**unresolved-ls**

```
1 v {
2 v   "info": {
3     "package": "https://github.com/oasis-tcs/openc2-ap-hunt",
4     "version": "0-wd01",
5     "title": "Unresolved ls schema",
6     "description": "Data definitions for Threat Hunting (TH) functions",
7     "namespaces": {
8       "ls": "http://oasis-open.org/openc2/oc2ls/v1.1"
9     },
10    "exports": ["Language-Spec-Types"]
11  },
12  "types": [
13    ["Language-Spec-Types", "Record", [], "", [
14      [1, "artifact", "ls_Artifact", "[[0"]], ""],
15      [2, "command", "ls_Command_ID", "[["0"]], ""],
16      [3, "device", "ls_Device", "[[0"]], ""
17    ]]
18  ]
19 }
```

**oc2ls-v1.1-lang\_resolved**

```
1 v {
2 v   "info": {
3     "package": "http://oasis-open.org/openc2/oc2ls/v1.1",
4     "title": "OpenC2 Language Profile",
5     "description": "Language Profile from the OpenC2 Language Specification version 1.1",
6     "exports": ["OpenC2-Command", "OpenC2-Response"]
7   },
8   "types": [
9     ["OpenC2-Command", "Record", [], "", [
10       [1, "action", "Action", [], ""],
11       [2, "target", "Target", [], ""],
12       [3, "args", "Args", "[[0"]], ""
13     ]]
14   ]
15 }
```

# Example Data Generation

Automatically create valid data instances given the JADN Schema

JADN Sandbox Home Creation ▾ Visualization Translation Validation Transformation Generation About

Home

 <b>Creation</b> Create and edit JADN schemas using forms, view JADN schemas in JSON Format. Create schema compliant data instances (documents, messages).	 <b>Schema Visualization</b> Convert a JADN Schema into different visual representations.	 <b>Schema Translation</b> Translate a JADN Schema to another Schema format. Translate a JSON Schema to a JADN Schema.
Schemas Data Instances	GraphViz HTML JIDL MarkDown PlantUML	JSON Relax (XML) XSD
 <b>Data Validation</b> Validate data instances in various data language formats against a JADN Schema.	 <b>Schema Transformation</b> Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc).	 <b>Example Data Generation</b> Generate various example data instances based off of a Schema.
CBOR JSON Relax (XML)		

Theme ▾ v0.10.0\_1705587211962

Go to: Generation

To Start : [click here](#)

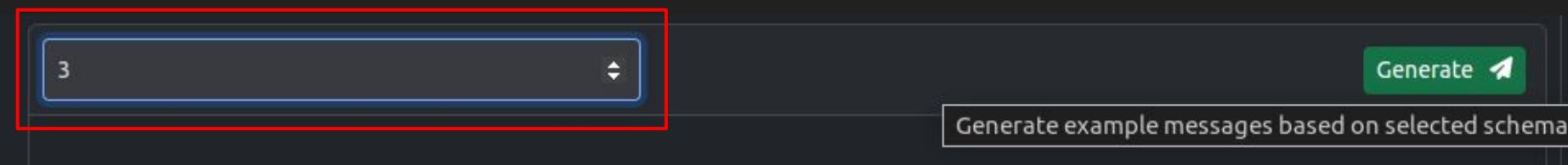
Note: Only a JADN Schema can be uploaded.

Enter the desired number of examples you would like generated. This number should be between 1 - 10.

(In this example, we have selected 3.)

After you input a number, click the green “Generate” button.

*Note: The button will only be clickable if you have a valid JADN schema and the number of examples desired.*



**Warning:** Example Generation may fail due to large Schema data. As a result, the application may lock itself. In this case, please restart docker container.

# Results

If successfully generated, you will be able to view and hide the example data generated.

For each example, you can:



- Download to local storage



- Save the file to data uploader  
(The file can then be found for easy access in the Data Validation page)



- Copy to clipboard

3

Data Example #1

```
1 v {  
2 v   "artist": {  
3 v     "artist_name": "consectetur magna dolor",  
4 v     "instruments": [  
5 v       [  
6 v         "harmonica",  
7 v         "brass",  
8 v         "keyboards",  
9 v         "guitar",  
10 v        "woodwinds",  
11 v        "guitar",  
12 v        "harmonica",  
13 v        "bass",  
14 v        "harmonica",  
15 v        "drums",  
16 v        "percussion",  
17 v        "drums",  
18 v        "vocals",  
19 v        "brass",  
20 v        "bass",  
21 v        "bass"
```

Data Example #2

```
1 v {  
2 v   "t_number": -4919358.7109660655,  
3 v   "title": "in",  
4 v   "length": "21:54:30.163Z",  
5 v   "featured": [  
6 v     [  
7 v       "artist_name": "magna nisi minim est",  
8 v       "instruments": [  
9 v         "
```

Data Example #3

```
1 v {  
2 v   "t_number": -4919358.7109660655,  
3 v   "title": "in",  
4 v   "length": "21:54:30.163Z",  
5 v   "featured": [  
6 v     [  
7 v       "artist_name": "magna nisi minim est",  
8 v       "instruments": [  
9 v         "
```

# Other Features

JADN Sandbox Home Creation Visualization Translation Validation Transformation Generation About

Home

**Creation**  
Create and edit JADN schemas using forms, view JADN schemas in JSON format.  
Create schema compliant data instances (documents, messages).

**Schemas Data Instances**

**Data Validation**  
Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Visualization**  
Convert a JADN Schema into different visual representations.

**Schema Translation**  
Translate a JADN Schema to another Schema format.

**Home**

**Creation**  
Create and edit JADN schemas using forms, view JADN schemas in JSON format.  
Create schema compliant data instances (documents, messages).

**Schemas Data Instances**

**Data Validation**  
Validate data instances in various data language formats against a JADN Schema.

CBOR JSON Relax (XML)

**Schema Visualization**  
Convert a JADN Schema into different visual representations.

**Schema Translation**  
Translate a JADN Schema to another Schema format.

**GraphViz HTML JIDL MarkDown PlantUML**

**JSON Relax (XML) XSD**

**Schema Transformation**  
Convert one or more JADN Schemas into a different but related Schema (resolve references, simplify by removing extensions, strip comments, etc.).

**Example Data Generation**  
Generate various example data instances based off of a Schema.

Theme ▾

Light  
Dark

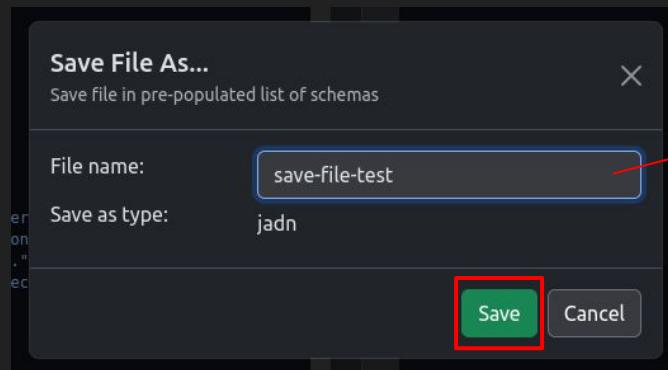
v0.9.1 1702496478525

**Theme Switcher: Choose between light and dark mode.**

Note: Dark is the default theme.



## Add custom files (Save as...)



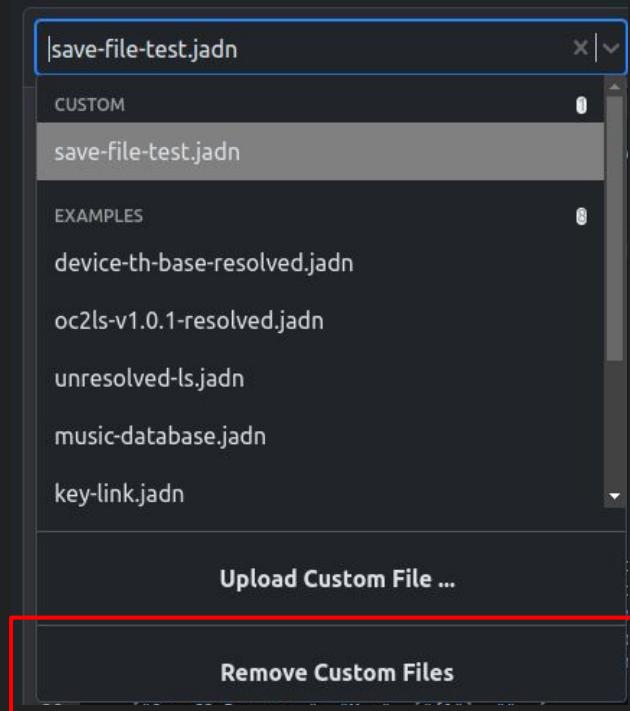
CUSTOM
save-file-test.jadn
EXAMPLES
device-th-base-resolved.jadn
oc2ls-v1.0.1-resolved.jadn
unresolved-ls.jadn
music-database.jadn
key-link.jadn

**Upload Custom File ...**

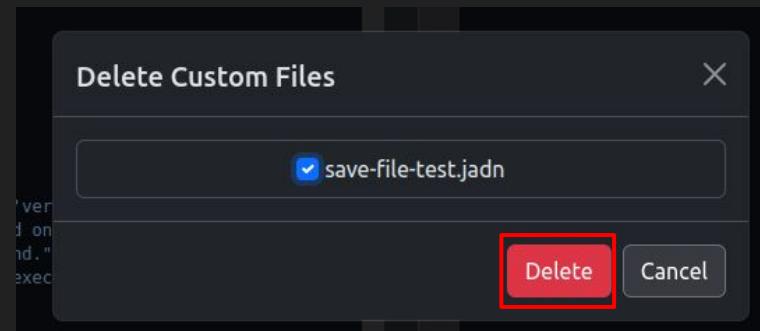
**Remove Custom Files**

The file saver will allow you to name and save data into a file to be found in the file loader for future use.

# Delete custom files

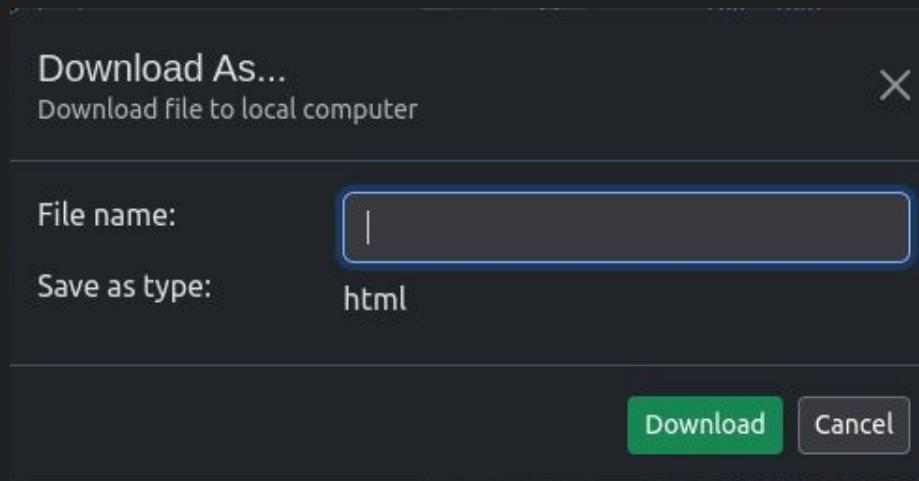


Custom files can be removed from the file loader if no longer needed.





## Download files to local storage (Download As..)



Files can be named and saved to your Downloads Folder.



# Docker Desktop Start Up Guide

JADN Sandbox

# Docker Table of Contents

[How to Get the JADN Sandbox Image](#)

[How to Run the JADN Sandbox Image](#)

- [From Images](#)<sup>1</sup>
- [From Containers](#)<sup>2</sup>
- [From Search Bar](#)

[How to Update the JADN Sandbox Image](#)

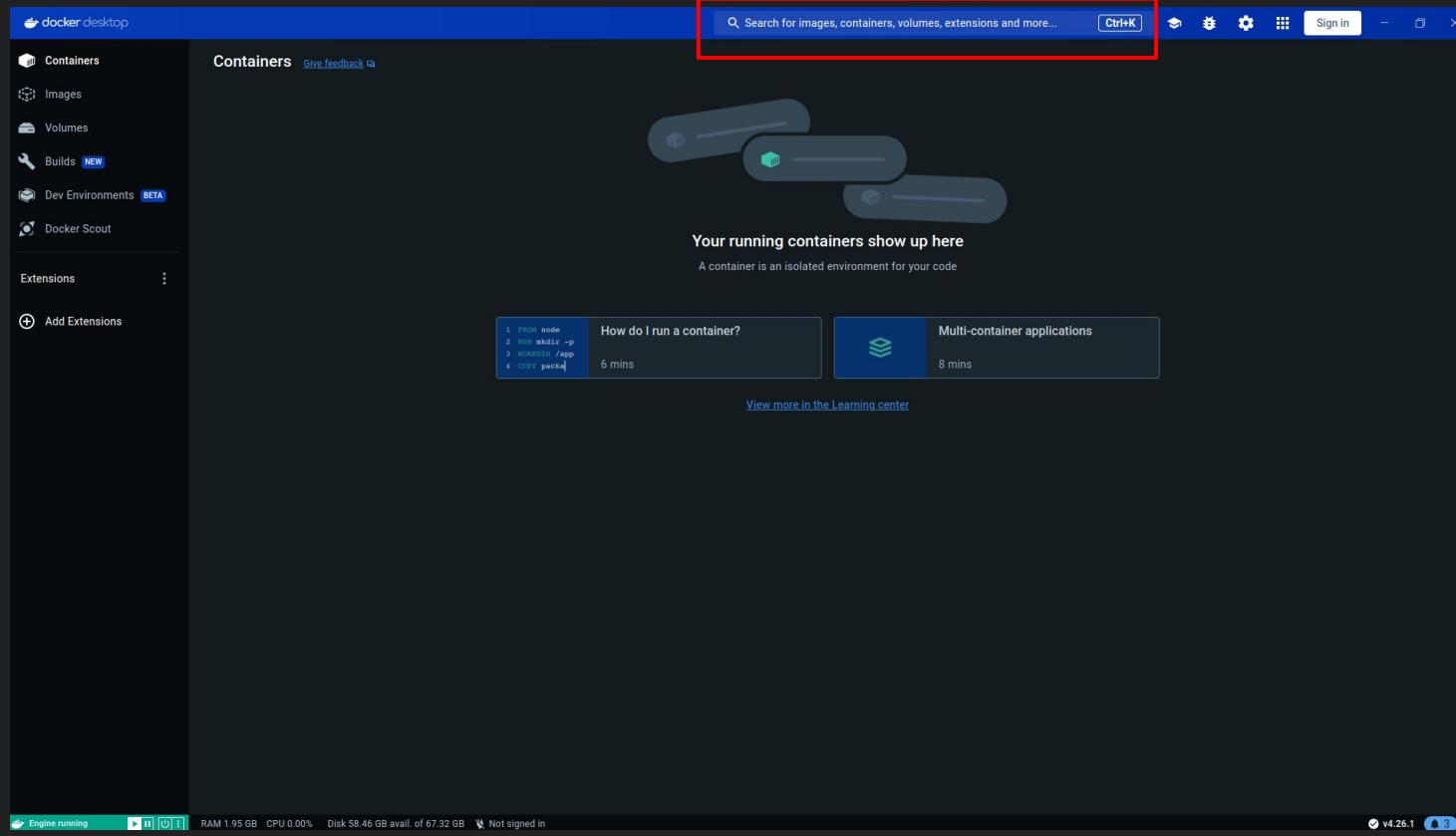
- [From Images](#)
- [From Search Bar](#)

[How to Stop the JADN Sandbox Image](#)

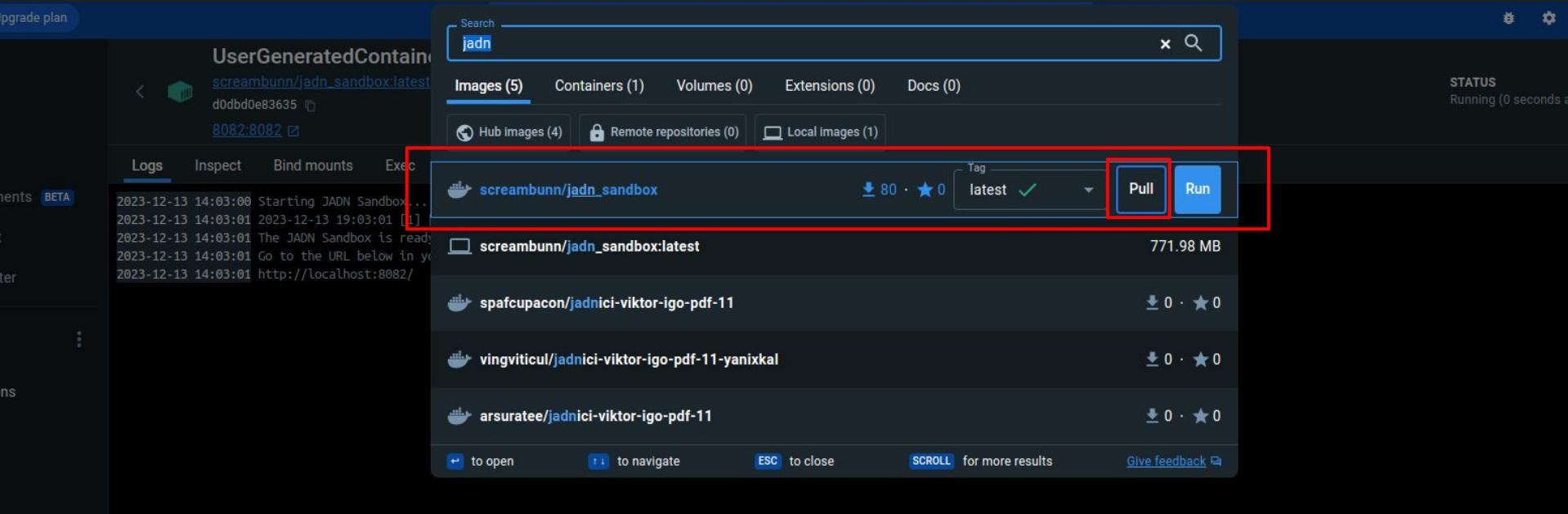
[How to Learn More about Docker](#)

1. A Docker image is like a set of instructions. It is the template loaded onto the container to run it.
2. A Docker container is a self-contained, runnable software application or service created from a docker image.

# How to Get the JADN Sandbox Image



Click in the search bar, search: jadn



Then, click Pull: screambunn/jadn\_sandbox

# How to Run the JADN Sandbox Image

[From Images](#)

[From Containers](#)

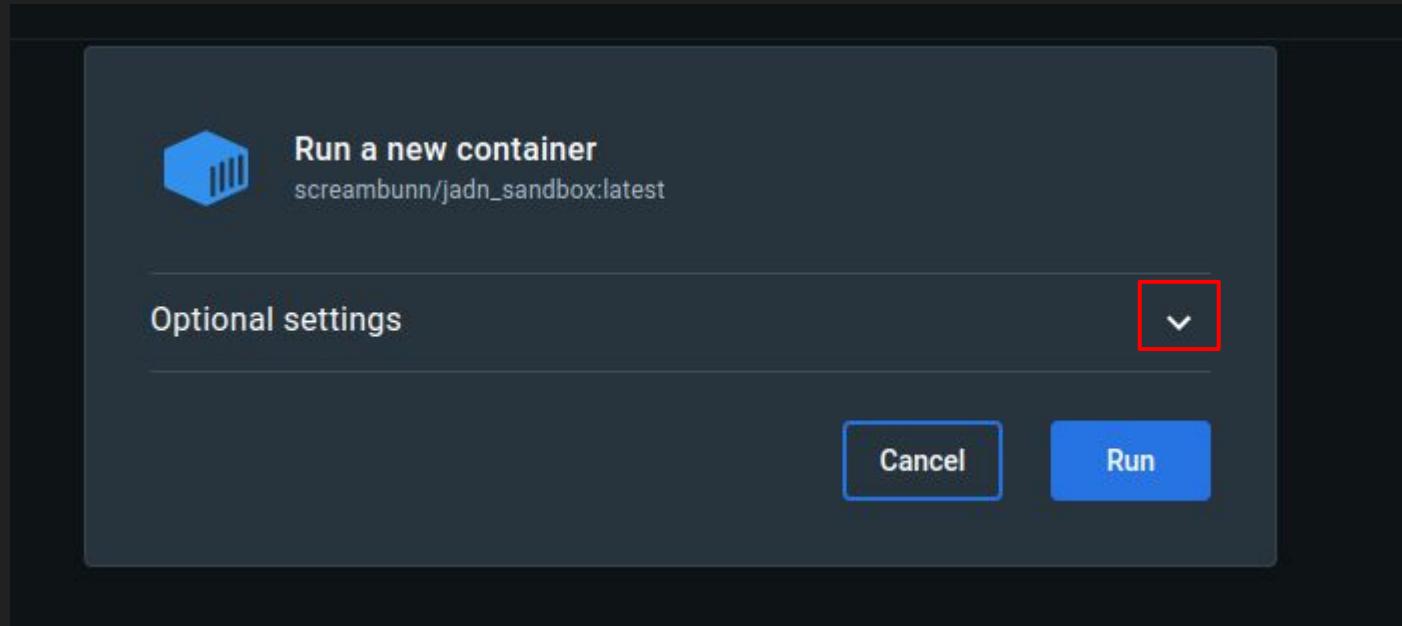
[From Search Bar](#)

# From Images

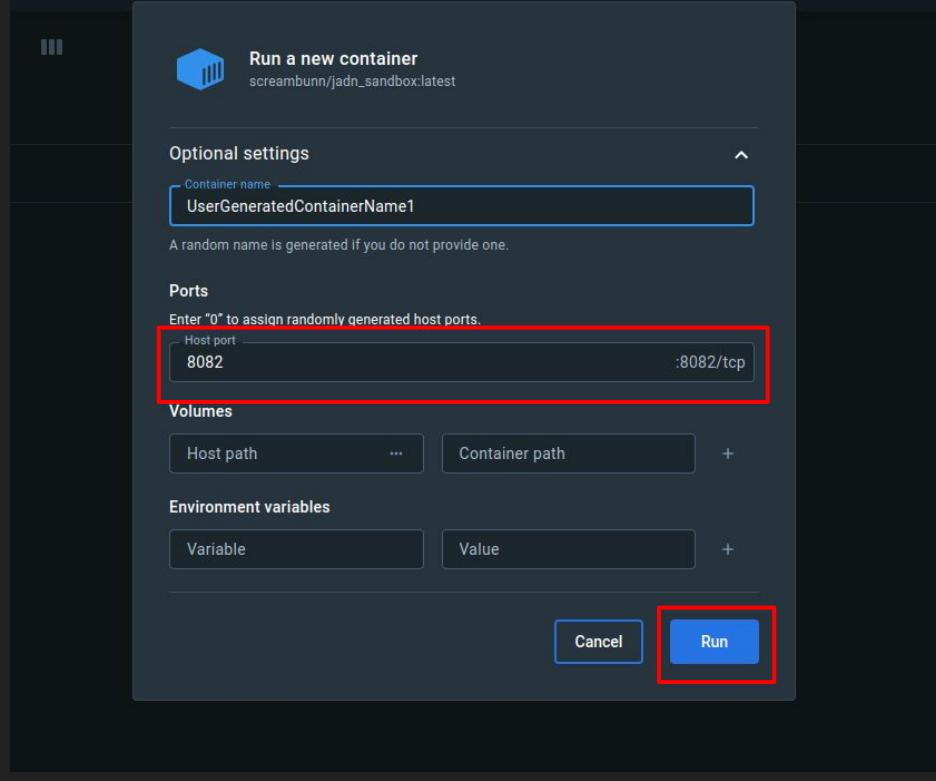
The screenshot shows the Docker Desktop interface. The left sidebar has options: Containers, **Images** (which is selected and highlighted with a red box), Volumes, Dev Environments (BETA), Docker Scout, Learning center, Extensions (with an 'Add Extensions' button), and a three-dot menu. The main area is titled 'Images' with a 'Give feedback' link. It shows 1 image in use (0 Bytes / 1.41 GB). A search bar and filter icons are at the top. Below is a table with columns: Name, Tag, Status, Created, Size, and Actions. One row is shown: 'screambunn/jadn\_sandbox' (tag latest, status Unused, created 22 hours ago, size 771.98 MB). The 'Actions' column for this row contains a play button (highlighted with a red box) and a 'Run' button.

Name	Tag	Status	Created	Size	Actions
screambunn/jadn_sandbox b6d3e6af3fb0	latest	Unused	22 hours ago	771.98 MB	

Under Images, find the desired image and click the play button to run the image.



After clicking the play button, this window will appear.  
You will need to enter information on the container to be able to run the image.  
Click the carrot icon (v) to expand the Optional settings.



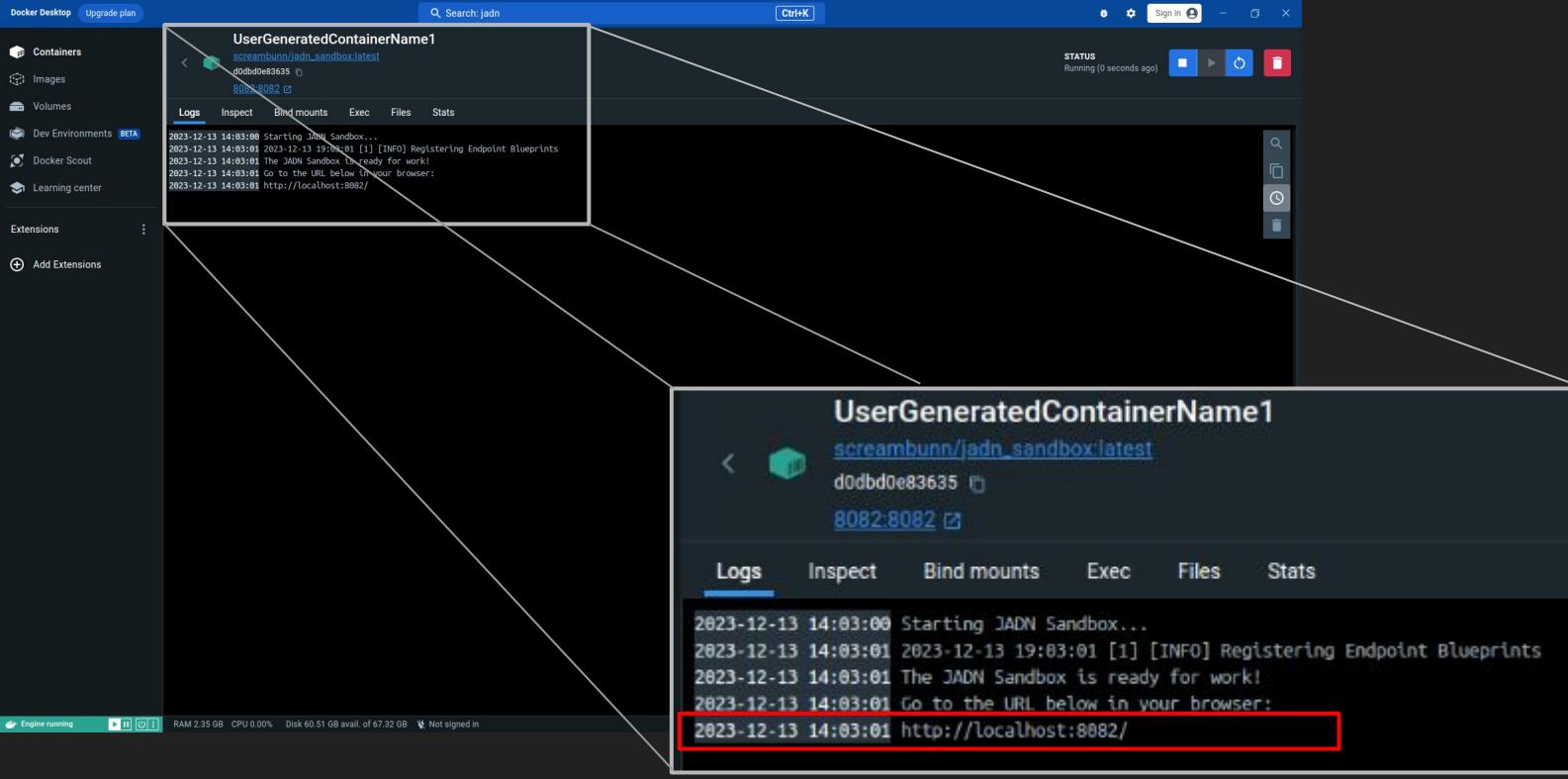
After clicking the carrot icon, you will see the settings as shown.

The Container Name is optional, but has been provided as an example (*UserGeneratedContainerName1*).

*Note: no spaces allowed in the container name.*

**REQUIRED INFORMATION:** Host Port - Enter: **8082**

Click Run.



After clicking on Run, a container will appear.

Click on the link provided to open the image in a browser window.

You can also enter the link itself in the browser:

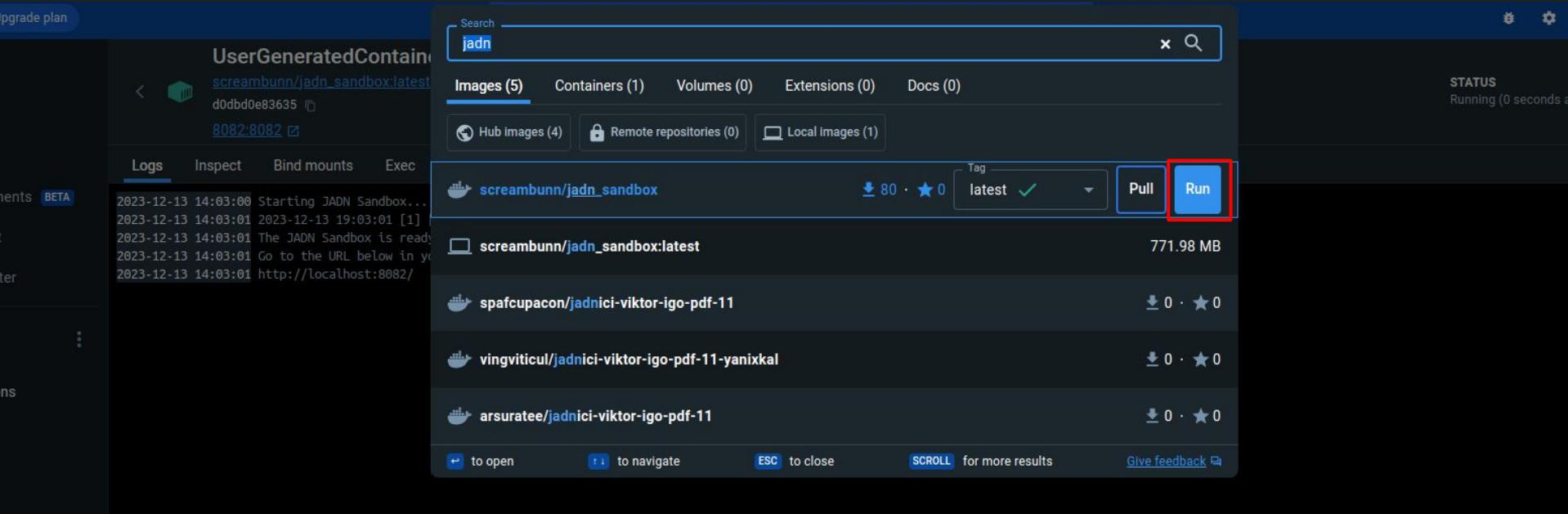
<http://localhost:8082/>

The screenshot shows the Docker Desktop application interface. On the left sidebar, there are icons for Containers, Images, Volumes, Dev Environments (BETA), Docker Scout (EARLY ACCESS), Learning center, Extensions, and a button to Add Extensions. The main area displays a container named "amazing\_brahmagupta" with the image "screambunn/jadn\_sandbox:latest". The container ID is "a3868fda2581" and it is running on port "8082:8082". The status is "Running (39 minutes ago)". A red box highlights the "Stop" button in the top right corner of the container card. Below the container card, there are tabs for Logs, Inspect, Bind mounts, Terminal, Files, and Stats. The Logs tab is selected, showing the following log output:

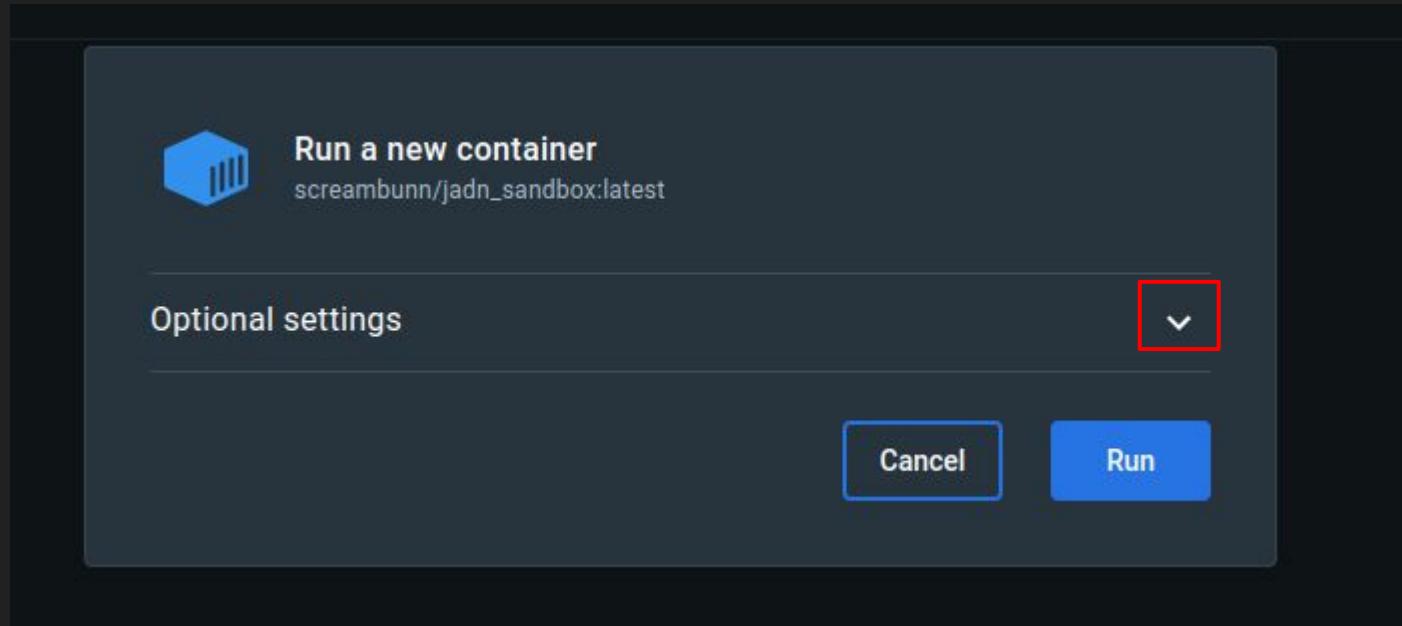
```
2024-02-07 14:28:23 Starting JADN Sandbox...
2024-02-07 14:28:23 2024-02-07 19:28:23 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 14:28:24 The JADN Sandbox is ready for work!
2024-02-07 14:28:24 Go to the URL below in your browser:
2024-02-07 14:28:24 http://localhost:8082/
```

To stop the container, click the stop button.

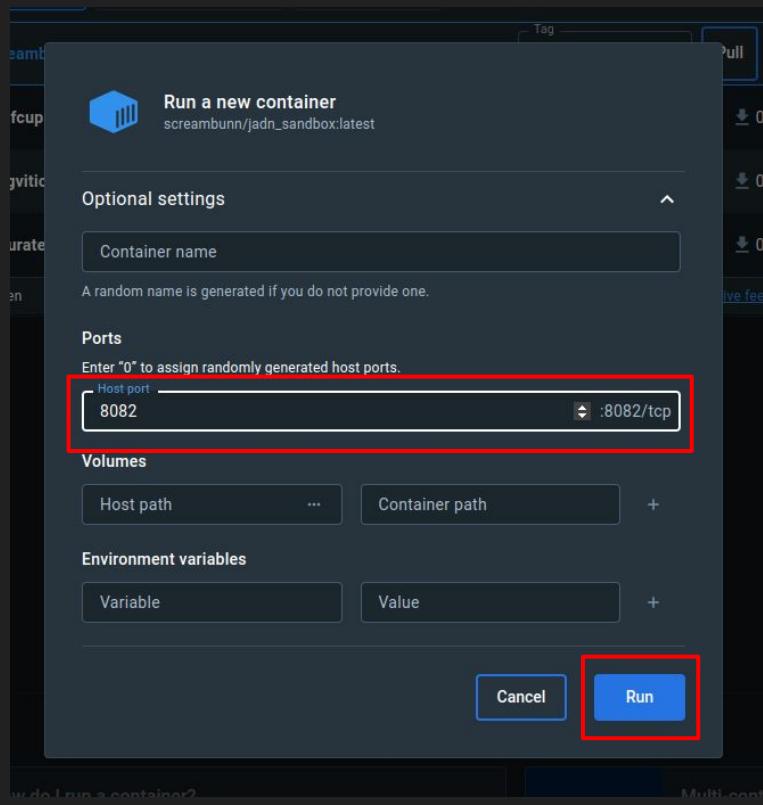
# From the Search Bar



In the search bar, search: jadn  
Then, click Run: screambunn/jadn\_sandbox



After clicking the play button, this window will appear.  
You will need to enter information on the container to be able to run the image.  
Click the carrot icon (v) to expand the Optional settings.



After clicking the carrot icon, you will see the settings as shown.

The Container Name is optional.

*Note: no spaces allowed in the container name.*

**REQUIRED INFORMATION:** Host Port - Enter: **8082**

Click Run.

Docker Desktop Update to latest

Search for images, containers, volumes, extensions and more... **⌘K**

Containers Images Volumes Dev Environments **BETA** Docker Scout **EARLY ACCESS** Learning center Extensions Add Extensions

**amazing\_brahmagupta**

screambunn/jadn\_sandbox:latest  
a3868fda2581 ⚡  
8082:8082 ⚡

STATUS  
Running (39 minutes ago)

Stop

Logs Inspect Bind mounts Terminal Files Stats

```
2024-02-07 14:28:23 Starting JADN Sandbox...
2024-02-07 14:28:23 2024-02-07 19:28:23 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 14:28:24 The JADN Sandbox is ready for work!
2024-02-07 14:28:24 Go to the URL below in your browser:
2024-02-07 14:28:24 http://localhost:8082/
```

After clicking on Run, a container will appear.

Click on the link provided to open the image in a browser window.

You can also enter the link itself in the browser:

<http://localhost:8082/>

# From Containers

Docker Desktop Upgrade plan

Q Search: jadn Ctrl+K

Containers Images Volumes Dev Environments BETA Docker Scout Learning center Extensions Add Extensions

Containers Give feedback

Container CPU usage 0.04% / 800% (8 cores allocated) Container memory usage 119.9MB / 3.62GB Show charts

Search Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
elegant_murdock 0592632c37e8	screambunn/jadn_sandbox:latest	Exited	0.04%	8082:8082	26 seconds ago	
stupefied_rosalind 6fe0cec2accd	screambunn/jadn_sandbox:latest	Running	0%	8082:8082	0 seconds ago	

The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. At the top, there are performance metrics for CPU usage (0.04% / 800%) and memory usage (119.9MB / 3.62GB). Below these are search and filter options. The main area displays a table of running containers. One container, 'stupefied\_rosalind', is highlighted with a red box around its status cell ('Running') and another red box around its stop button in the actions column.

First, make sure to stop any running containers.

Docker Desktop Upgrade plan

Search: jadn Ctrl+K

Containers Give feedback

Container CPU usage ⓘ Container memory usage ⓘ Show charts ▾

No containers are running.

Search Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
elegant_murdock 0592632c37e8	screambunn/jadn_sandbox:latest	Exited	N/A	8082:8082	2 minutes ago	
stupefied_rosalind 6fe0cec2accd	screambunn/jadn_sandbox:latest	Exited	N/A	8082:8082	4 seconds ago	

Containers Images Volumes Dev Environments BETA Docker Scout Learning center Extensions Add Extensions

Click play on the desired container and then click on link to see terminal of running container.

Docker Desktop Upgrade plan

elegant\_murdock

screambunn/jadn\_sandbox:latest

0592632c37e8

8082:8082

STATUS  
Running (32 seconds ago)

Logs Inspect Bind mounts Exec Files Stats

```
2023-12-21 09:36:10 Starting JADN Sandbox...
2023-12-21 09:36:10 2023-12-21 14:36:10 [1] [INFO] Registering Endpoint Blueprints
2023-12-21 09:36:10 The JADN Sandbox is ready for work!
2023-12-21 09:36:10 Go to the URL below in your browser:
2023-12-21 09:36:10 http://localhost:8082/
2023-12-21 09:38:58 Starting JADN Sandbox...
2023-12-21 09:38:58 2023-12-21 14:38:58 [1] [INFO] Registering Endpoint Blueprints
2023-12-21 09:38:58 The JADN Sandbox is ready for work!
2023-12-21 09:38:58 Go to the URL below in your browser:
2023-12-21 09:38:58 http://localhost:8082/
2023-12-21 09:41:35 Starting JADN Sandbox...
2023-12-21 09:41:35 2023-12-21 14:41:35 [1] [INFO] Registering Endpoint Blueprints
2023-12-21 09:41:35 The JADN Sandbox is ready for work!
2023-12-21 09:41:35 Go to the URL below in your browser:
2023-12-21 09:41:35 http://localhost:8082/
```

Click on the link provided to open the image in a browser window.

You can also enter the link itself in the browser:

<http://localhost:8082/>

# How to Update the JADN Sandbox Image

[From Images](#)

[From Search Bar](#)

# From Images

Docker Desktop Upgrade plan

Containers [Give feedback](#)

Container CPU usage ⓘ

No containers are running.

Container memory usage ⓘ

No containers are running.

Show charts ▾

Ctrl+K

b6d3e6af3fb0

Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
kind_cori f5070dd74397	screambunn/jadn_sandbox:latest	Exited	N/A	8082:8082 ↗	31 seconds ago	<a href="#">▶</a> <a href="#">⋮</a> <a href="#">Delete</a>

**OPTIONAL:** Delete all containers related to the image (screambunn/jadn\_sandbox:latest) by clicking the delete button (trash can icon) under Action.

*Note: Containers use the image that it was created with. Even though the image is named the same, its versioning is different. Therefore older containers will be out of date.*

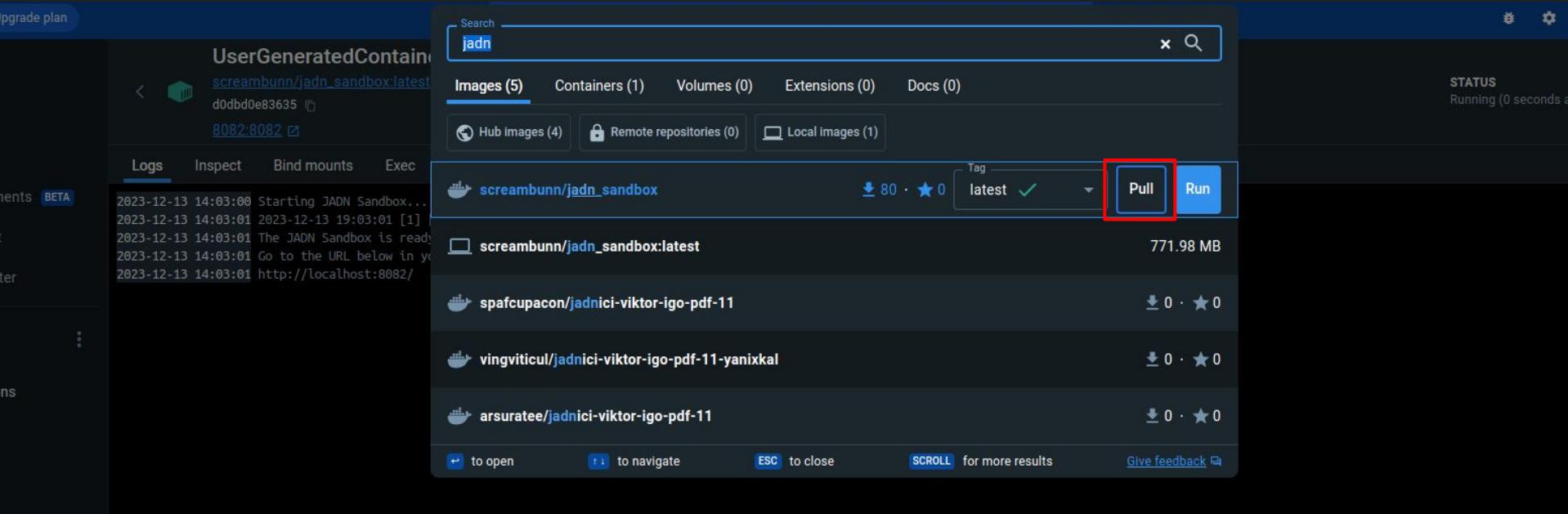
The screenshot shows the Docker Desktop interface. The left sidebar includes options like Containers, Images (which is selected and highlighted with a red box), Volumes, Dev Environments (Beta), Docker Scout, Learning center, Extensions, and Add Extensions. The main area is titled 'Images' with a search bar and a 'Ctrl+K' keyboard shortcut. It displays a table of images with columns for Name, Tag, Status, Created, Size, and Actions. A single image entry is shown: 'screambunn/jadn\_sandbox' (tag latest) - Unused, created 22 hours ago, size 771.93 MB. A context menu is open over this entry, also highlighted with a red box, showing options: 'View packages and CVEs', 'Pull' (selected and highlighted with a red box), and 'Push to Hub'.

Name	Tag	Status	Created	Size	Actions
screambunn/jadn_sandbox	latest	Unused	22 hours ago	771.93 MB	<ul style="list-style-type: none"><li>▶</li><li>⋮</li><li>✖</li><li><small>View packages and CVEs</small></li><li><small>Pull</small> (Selected)</li><li><small>Push to Hub</small></li></ul>

Under Images, Go to Actions and click the 3 dotted icon.  
A menu drop down will appear. Click Pull.  
Then go through the [steps](#) to run the image.

Pulling image..

# From the Search Bar



In the search bar, search: jadn  
Then, click Pull: screambunn/jadn\_sandbox

# How to Stop the JADN Sandbox Image

Docker Desktop Upgrade plan

Search for images, containers, volumes, extensions and more... Ctrl+K

Containers Give feedback ⓘ

Container CPU usage ⓘ Container memory usage ⓘ Show charts ▾

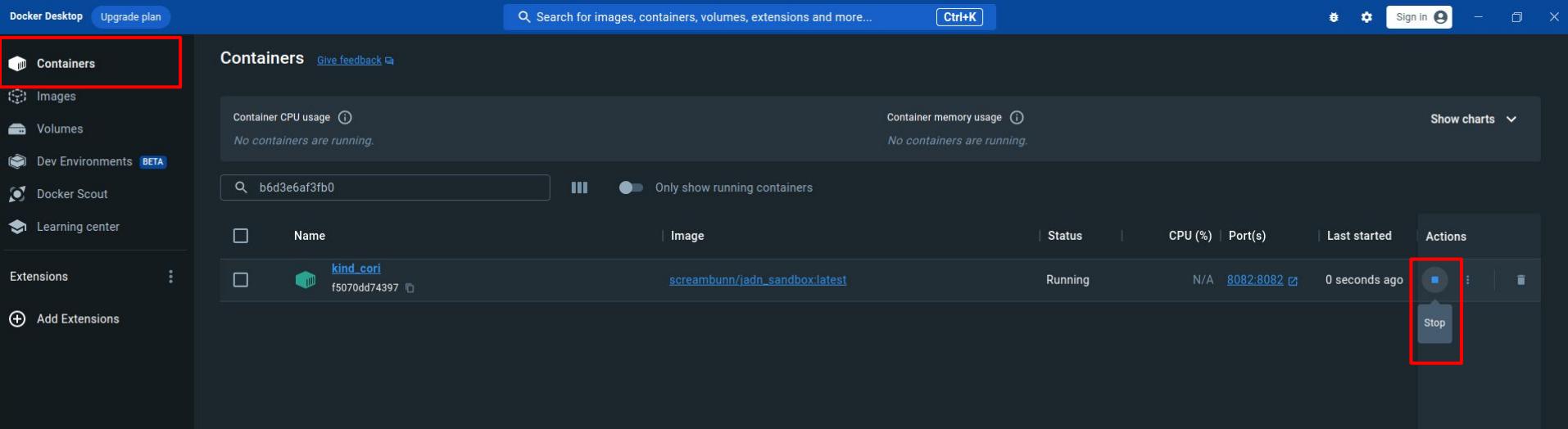
No containers are running.

b6d3e6af3fb0

Only show running containers

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
kind_cori f5070dd74397	screambunn/jadn_sandbox:latest	Running	N/A	8082:8082	0 seconds ago	 Stop

Add Extensions



Click the stop button under Action in Containers.

Containers

Images

Volumes

 Dev Environments BETA Docker Scout EARLY ACCESS

Learning center

Extensions

Add Extensions

## sad\_ritchie

[screambunn/jadn\\_sandbox:latest](#)

00bd8f6577d3

STATUS

Running (0 seconds ago)

[Logs](#) [Inspect](#) [Bind mounts](#) [Terminal](#) [Files](#) [Stats](#)

```
2024-02-07 15:21:40 Starting JADN Sandbox...
2024-02-07 15:21:41 2024-02-07 20:21:41 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 15:21:41 The JADN Sandbox is ready for work!
2024-02-07 15:21:41 Go to the URL below in your browser:
2024-02-07 15:21:41 http://localhost:8082/
2024-02-07 15:21:48 Starting JADN Sandbox...
2024-02-07 15:21:48 2024-02-07 20:21:48 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 15:21:48 The JADN Sandbox is ready for work!
2024-02-07 15:21:48 Go to the URL below in your browser:
2024-02-07 15:21:48 http://localhost:8082/
2024-02-07 15:21:52 Starting JADN Sandbox...
2024-02-07 15:21:52 2024-02-07 20:21:52 [1] [INFO] Registering Endpoint Blueprints
2024-02-07 15:21:52 The JADN Sandbox is ready for work!
2024-02-07 15:21:52 Go to the URL below in your browser:
2024-02-07 15:21:52 http://localhost:8082/
```



Alternatively, Click the stop button within the running container

# How to Learn More about Docker

- Containers
- Images
- Volumes
- Dev Environments BETA
- Docker Scout
- Learning center**

## Learning center [Give feedback](#)

### Walkthroughs

Quick hands-on guides to show you around

```
1 FROM node  
2 RUN mkdir -p  
3 WORKDIR /app  
4 COPY package.json .
```

#### How do I run a container?

6 mins



#### Multi-container applications

6 mins

[View all](#)

### AI/ML guides

Get started with AI/ML using Docker



#### GenAI Stack

Start your GenAI application using Neo4j, Langchain, Ollama, Python, and Docker Compose

### Docker for beginners by language

45 min guides written for different programming languages



NodeJS



Python



Go



Java



C# (.NET)



Rust

[Request a guide](#)

### Samples

Go to the Learning center to learn more about Docker.