Process Guide: JSON Generation for Furniture Manufacturing

This guide describes the complete process for transforming technical drawing data into a structured JSON file for furniture manufacturing. The resulting JSON will be used by an Adobe Illustrator script to create manufacturing posters with templates, holes, labels, symbols, and connection areas.

Process Overview

INPUT → PROCESSING → OUTPUT
Technical Drawing → Generation System → Structured JSON

Detailed Step-by-Step Process

Step 1: Input Data Preprocessing

Convert all measurements to millimeters

- If dimensions are in centimeters, multiply by 10
- Example: 45 cm → 450 mm

Round dimensions and coordinates

- Round to the first decimal place
- If the value is within ±0.1 mm of a whole number, round to that integer
- Example: 299.87 mm → 299.9 mm → 300.0 mm; 19.99 mm → 20.0 mm → 20.0 mm

Step 2: Define Piece Dimensions and Orientation

Determine fundamental dimensions

- **height**: The largest dimension of the piece
- **length**: The intermediate dimension
- **thickness**: The smallest dimension

Calculate half thickness

- half_thickness = thickness / 2
- Apply the same rounding rule
- Example: thickness = 20.0 mm → half_thickness = 10.0 mm

Establish coordinate system for each face (Person Metaphor)

- Consider the piece vertically, like a standing person
- **Main face (main)**: length × height, origin (0,0) at bottom left corner

- **Opposite main face (other_main)**: Opposite to main
- **Top face (top)**: length × thickness, like the person's "head"
- **Bottom face (bottom)**: length × thickness, like the person's "feet"
- **Left face (left)**: thickness × height, like the person's "left arm"
- **Right face (right)**: thickness × height, like the person's "right arm"

Step 3: Map Pieces in 3D Space

Create bounding boxes

- Combine information from three views (top, side, front)
- Define X, Y, Z coordinates for each piece in space

Determine relative positions

- Check overlaps between pieces on X, Y, and Z axes
- Identify intersection points between pieces

Step 4: Allocate Initial Objective Holes

For each piece, define temporary holes on main faces

- **Main and other_main faces**: flap_corner holes at four corners
- Coordinates: (half_thickness, half_thickness), (half_thickness, height-half_thickness), etc.
- **Top, bottom, left and right faces**: top_corner holes at corners
- Coordinates: (half_thickness, half_thickness), (length-half_thickness, half_thickness)

Add intermediate holes when necessary

- If distance between two holes is greater than 200 mm, add intermediate holes dividing the resulting space
- Maintain homogeneous hole distribution
- Respect correct classification of each hole:
  - Between two flap_corner: add flap_central
  - Between two top_corner: add top_central

Step 5: Infer Connections Between Pieces

Identify overlap areas between pieces at any angle

- Check overlap on X, Y, and Z axes to detect connections in all possible directions
- Identify lateral, vertical, or any-angle connections
- Minimum overlap of 10 mm on corresponding axis to consider a valid connection

Calculate intersection points

- Determine intersection limits between pieces: (X_min, X_max, Y_min, Y_max)
- Use these points as reference for hole positioning

Assign connection IDs

- Each connection between two pieces receives a unique ID
- All holes associated with a connection share the same connectionId

Step 6: Map Holes Between Connected Pieces

For each identified connection

- Map the corresponding face of the primary piece on the secondary piece

Define connection area

- Allocate subjective holes paired with objective holes already existing on other pieces
- Ensure perfect alignment with the primary hole of the other piece

Validate and classify mapped holes

- Verify coordinates are within piece limits
- Classify holes based on position:
  - **flap_corner**: At face corners (main or other_main), at half_thickness from both edges
  - **flap_central**: At main or other_main face edges, at half_thickness from one edge
  - **face_central**: In the middle of the face (not at edges)
  - **top_corner**: In thickness (top, bottom, left or right), at corners
  - **top_central**: In thickness (top, bottom, left or right), centered between other holes
  - **singer_flap**: Diagonal hole near half_thickness from edge
  - **singer_central**: Diagonal hole in the middle of the face
  - **singer_channel**: Diagonal hole near half_thickness from two parallel edges. Used more in slats

Clean unconnected holes

- Remove temporary holes that are not part of a valid connection
- Keep only holes with assigned connectionId

Step 7: Add Reinforcement Holes (Singer)

For face-to-top connections

- Add singer holes on the face opposite to the connected face
- Mirror coordinates to the opposite face
- Do not assign connectionId to singer holes

Classify singer holes

- **singer_flap**: Near edges
- **singer_central**: In the middle of the face, at least 50 mm from edges
- **singer_channel**: At specific junction positions

Step 8: Choose Model Template

Count holes by thickness type

- Map quantity of top holes (top_corner and top_central) grouped by thickness

Select template

- Choose the thickness with the most top holes
- In case of tie, use the smallest value (17, 20, 25, 30)

Adjust incompatible holes

- For pieces with thickness greater than chosen template:
  - Replace top holes with flap holes on corresponding face
  - Maintain the model template's targetType

Step 9: Define Hardware for Each Hole

Assign hardware based on hole type

- **flap_corner, flap_central, face_central**: dowel_M_with_glue
- **top_corner, top_central**: glue
- **singer_flap, singer_central, singer_channel**: dowel_G_with_glue, no_limiter

Define diameter and depth

- Dowel holes: 8 mm diameter
- Face holes: 10 mm depth
- Thickness holes: 20 mm depth
- Singer holes: No limiter, 40 mm depth

Step 10: Create Connection Areas

For each identified connection

On primary piece (e.g., top):

- Create rectangular area based on effectively overlapped/connected area
- Respect real limits of overlap area
- Use fill: "black" and opacity: 0.05
- Associate with same connectionId of corresponding holes

On secondary piece (e.g., leg):

- Replicate connection area on corresponding face
- Respect same effective overlap area
- Same connectionId

Validate area limits

- Ensure coordinates are within face limits
- Adjust if necessary

Step 11: Structure Final JSON

Create main structure

```json
{
  "pieces": [
    {
      "name": "Piece Name",
      "length": 450,
      "height": 100,
      "thickness": 20,
      "quantity": 1,
      "faces": [ ... ]
    },
    ...
  ]
}
```

For each piece, add all faces

```json
```

```
"faces": [
  {
    "faceSide": "main",
    "holes": [ ... ],
    "connectionAreas": [ ... ]
  },
  {
    "faceSide": "other_main",
    "holes": [ ... ],
    "connectionAreas": [ ... ]
  },
  ...
]
```

For each face, add holes and connection areas

```json
"holes": [
  {
    "x": 10,
    "y": 10,
    "type": "flap_corner",
    "targetType": "20",
    "ferragemSymbols": ["dowel_M_with_glue"],
    "connectionId": 1
  },
  ...
],
"connectionAreas": [
  {
    "x_min": 0,
    "y_min": 0,
    "x_max": 200,
    "y_max": 20,
    "fill": "black",
    "opacity": 0.05,
    "connectionId": 1
  },
  ...
]
```

Validate resulting JSON

- Verify all required fields are present

- Confirm coordinates and dimensions are within expected limits
- Ensure all related holes share the same connectionId

Considerations for Python Implementation

Recommended data structures

- Use classes to represent Pieces, Faces, Holes, and Connection Areas
- Use dictionaries for connection mapping

Main algorithms

- Implement custom rounding function
- Develop algorithm for 3D intersection detection
- Create function for hole mapping between pieces

Important validations

- Check coordinate limits
- Validate coherence between holes and their connections
- Confirm all pieces have at least one valid connection

This guide serves as the foundation for developing a Python system that automatically performs the entire process of transforming technical drawing data into structured JSON for furniture manufacturing.