Templates

> This document is written in Markdown.
> If it looks unstyled here, view it on GitHub for a formatted version.

Templates let you define what fields appear and in what order. They are plain JSON files you can edit in any text editor. Templates control layout only. They cannot change app behavior or data.:contentReference[oaicite:9]{index=9}

1) The mental model (two modes)

Dataset mode

Dataset mode is for big flat arrays (table-like). Templates do not apply in this mode.:contentReference[oaicite:10]{index=10}

Records mode

Records mode is for structured objects and API responses. Templates control layout and record labels here.:contentReference[oaicite:11]{index=11}

If you are not seeing template effects, first check your mode.

2) The basic workflow

1. In JTF, click **Download** in the Templates panel to get a starter template.
2. Edit it (paths and labels).
3. Upload the template back into JTF.
4. Select it in the Template dropdown.

Tip: If your browser shows the JSON instead of downloading it, use Save As and store it locally.:contentReference[oaicite:12]{index=12}:contentReference[oaicite:13]{index=13}

3) Template anatomy (what's inside)

A template contains:

- `templateName`
- Optional `match` rules (controls when it applies)
- Optional `recordLabel` rules (controls the Record dropdown label)
- `layout`: sections and fields (controls what you see)

If a path does not exist, it is skipped. Nothing crashes.:contentReference[oaicite:14]{index=14}

4) Paths: how JTF finds your values

Paths are dot-separated:

- `Status.Name`
- `AssignedTo.Email`
- `ViewerUrl`

JTF reads the record and walks the path. If anything is missing along the way, the field does not render.

Practical tip:

- Start by copying real property names from the Raw JSON view.
- Build one field at a time.

5) Layout: sections and fields

`layout` is an ordered list of sections. Each section has an ordered list of fields.

That is the whole trick.

Example shape:

- Section: "Header"
  - Field: path + label + format
- Section: "Details"
  - Field: path + label + format

6) Formats (how values display)

Common formats used in templates:

- `text`: default
- `badge`: compact label pill (great for status)
- `date`: readable date/time
- `link`: clickable URL
- `multiline`: preserves line breaks
- `json`: shows a nested object (can be collapsible)

If a value is the wrong type for a format, the field is skipped.

7) Match rules (what they do, and what they do not)

Match rules decide whether a template applies to a given record.

Match is not filtering. Match is not a query language. Match does not transform data.

Keep match rules simple so templates stay predictable.

8) Record labels (the Record dropdown)

A template can build record labels from multiple fields in order. If fields are missing, JTF falls back safely.

Use this to make “Record 12” become “#RFI-1042 Window detail (Open)”.

9) When a template matches but shows nothing

Sometimes the template technically matches, but none of its paths exist on that record.

JTF will show a friendly hint instead of a blank viewer, and you still have access to the full record view.:contentReference[oaicite:15]{index=15}

Fix is usually one of:

- Wrong mode (Dataset vs Records)
- Wrong path (typo or different casing)
- Template is for a different record shape

10) Guardrails (important)

Templates are declarative only. No scripting. No conditional logic. No data mutation. JTF reads data. It does not change data.:contentReference[oaicite:16]{index=16}