

JSON Tool

- [Introduction](#)
- [JSON Tool Properties](#)
 - [Configure JSON Tool](#)
 - [Store To Form](#)
 - [Store To Workflow Variable](#)
- [Notes On JSON Returned Data](#)
- [Download Demo App](#)
- [Related Documentation](#)

Introduction

The **JSON Tool** in a process enables one to issue a JSON web service call, and to save the returned data into Joget's form data and/or into the process's workflow variable. [Download the demo app](#) to try it out on your Joget DX platform.

The Joget Marketplace has a free [JSON Form Options Plugin](#) to use JSON to populate Form Select Box, Check Box or Radio Buttons.

The screenshot displays the Joget DX interface for configuring a process. On the left is a sidebar with navigation options: Design App (test1), Forms & UI, Processes (selected), Properties & Export, Performance, Logs, and All Apps. The main area shows the 'Process name' as 'Process 1' and a workflow diagram with a single tool named 'Tool 1'. Below the diagram, the 'Show Additional Info' section is active, showing tabs for 'Map Tools to Plugins' (selected), 'Map Routes to Plugins', and 'Variable List'. A table lists the tools defined in the workflow design, with 'Tool 1' (ID: tool1) highlighted. The table has columns for 'Plugin Name' (Json Tool) and 'Plugin Version' (5.0.0). Below the table are buttons for 'Remove Plugin' and 'Configure Plugin'. A red circle highlights the 'Plugin Name' and 'Plugin Version' fields. A red arrow points from the 'Add/Edit Plugin' button to the 'Configure Plugin' button.

Design App: test1
Version 1 Not Published

Forms & UI
Processes
Properties & Export
Performance
Logs
All Apps

Process name: Process 1

Participant 1

Tool 1

Show Additional Info

Map Participants to Users Map Activities to Forms **Map Tools to Plugins** Map Routes to Plugins Variable List

This is the list of tools defined in the Workflow design.

Q

Tool 1
ID : tool1

Plugin Name	Json Tool
Plugin Version	5.0.0

Remove Plugin Configure Plugin

Add/Edit Plugin

JSON Tool Properties

Configure JSON Tool

Configure JSON Tool ?

JSON URL *

Call Type

GET ▼

Request Headers

NAME	VALUE
+	

No Response Expected

☐

Debug Mode ?

☐

Figure 1: Configure JSON Tool

Name	Description
JSON URL	URL to be called.
Call Type	<p>Select the call type:</p> <ul style="list-style-type: none"> GET POST <p>GET requests include all required data in the URL. GET is less secure compared to POST because data sent is part of the URL. So it's saved in browser history and server logs in plaintext.</p> <p>In contrast, HTTP POST requests supply additional data from the client (browser) to the server in the message body. POST is a little safer than GET because the parameters are not stored in browser history or in web server logs. From here.</p>
POST Method (Call type = POST)	<p>Select the post method:</p> <ul style="list-style-type: none"> POST Parameters POST Parameters as JSON Payload Custom JSON Payload
POST Parameters (Call type = POST)	<p>When POST Method is set to "POST Parameters", these parameters will be sent as a <code>UrlEncodedFormEntity</code>.</p> <p>When POST Method is set to "POST Parameters as JSON Payload", these parameters will be sent as a <code>StringEntity</code> in a form of an escaped JSON string.</p>
Custom JSON Payload	<p>Write your own JSON to be the payload. It will be sent as a <code>StringEntity</code>.</p> <p>This option is available only when "Custom JSON Payload" in selected.</p>
Request Headers	Add name(s) and value(s) to the request header.
No Response Expected	<p>Check if no response is expected, so that even if there is a response, this tool will simply ignore it.</p> <p>Using this option will also disable "store to form" and "store to workflow variable" properties.</p>
Debug Mode	Show relevant debug entries in the server log for debugging purposes.

Store To Form

Store To Form

Form

Base JSON Object Name for Multirow Data

Field Mapping

FIELD NAME

JSON OBJECT NAME

Figure 2: Store to Form

Name	Description						
Form	Select target form to store data to.						
Base JSON Object Name for Multirow Data	Name of the object that contains an array to be based on.						
Field Mapping	Mapping with JSON data with Form fields. <table><tr><th>Name</th><th>Description</th></tr><tr><td>Field Name</td><td>Form field ID</td></tr><tr><td>JSON Object Name</td><td>JSON property name</td></tr></table>	Name	Description	Field Name	Form field ID	JSON Object Name	JSON property name
Name	Description						
Field Name	Form field ID						
JSON Object Name	JSON property name						

Store To Workflow Variable

Store To Workflow Variable

Workflow Variable Mapping

WORKFLOW VARIABLE

JSON OBJECT NAME

Figure 3: Store to Workflow Variable

Name	Description						
Workflow Variable Mapping	<table><tr><th>Name</th><th>Description</th></tr><tr><td>Workflow Variable</td><td>Workflow Variable Name.</td></tr><tr><td>JSON Object Name</td><td>JSON property name.</td></tr></table>	Name	Description	Workflow Variable	Workflow Variable Name.	JSON Object Name	JSON property name.
Name	Description						
Workflow Variable	Workflow Variable Name.						
JSON Object Name	JSON property name.						

Notes On JSON Returned Data



Introduced in v5, the API Domain Whitelist setting in [General Settings](#) needs to be configured to allow JSON API requests. If a request is from a non-whitelisted domain, the response will be a **HTTP 400 Bad Request**. Enter asterisk "*" into the **API Domain Whitelist** field in [General Settings](#) to allow API calls.

In figure 2 and 3 above, you can specify how to treat the returned data. The returned data may be saved as form data or/and to be saved into process's workflow variable. The example used in this article shows how one can store multi-row data into a form data table.

Sample JSON API **POST** call: <http://localhost:8080/jw/web/json/apps/published/userviews>

Sample JSON Returned Results

```
{
  "apps": [
    {
      "name": "App Center",
      "userviews": [
        {
          "name": "Joget DX",
          "id": "v",
          "version": 1,
          "url": "/jw/web/userview/appcenter/v"
        },
        {
          "name": "Joget DX Platform",
          "id": "v2",
          "version": 1,
          "url": "/jw/web/userview/appcenter/v2"
        }
      ],
      "id": "appcenter",
      "version": 1
    },
    {
      "name": "Customer Relationship Management",
      "userviews": [
        {
          "imageUrl": "/jw/web/app/crm/resources/crm_icon.png",
          "name": "Customer Relationship Management",
          "id": "crm_userview_sales",
          "version": 1,
          "url": "/jw/web/userview/crm/crm_userview_sales"
        }
      ],
      "id": "crm",
      "version": 1
    }
  ]
}
```

[Download Demo App](#)

JSON TOOL DX DEMO APP

View JSON Records 2

Run Process JSON Userviews API

Admin Admin

Home > View JSON Records

<input type="checkbox"/>	ID	APP NAME	APP VERSION	
<input type="checkbox"/>	appcenter	App Center	1	Edit
<input type="checkbox"/>	crm	Customer Relationship Management	1	Edit

2 items found, displaying all items.

[CSV](#) | [Excel](#) | [XML](#) | [PDF](#)

Delete

Figure 4: Download the demo app below to view how JSON TOOL is used in run process to populate form records

- [APP_json_tool_dx_kb.jwa](#)

Related Documentation

- [List of JSON API](#)
- [JSON Form Options Plugin](#) From Joget Marketplace
- [Configure JSON Tool Based On Returned JSON Data Structure](#)
- [Sample JSON API Integration](#)
- [Single Sign On - SSO](#)