

Using Excel files as sources for ConfiForms fields

Since version 1.53.4 we have introduced couple of **experimental APIs** which help you to make a bridge between ConfiForms web-services backed fields and Excel files attached to your Confluence page.

3 services:

- excel-2json
- excel2007-2json
- csv-2json (starting from ConfiForms 2.27.17 - [Release Notes#Version2.27.17](#))

These 2 APIs are available at the following URLs:

File format	API URL	Method
Excel 97-2004 (xls)	https://YOUR_CONFLUENCE_SERVER/ajax/confiforms/rest/excel-2json.action	GET
Excel (xlsx)	https://YOUR_CONFLUENCE_SERVER/ajax/confiforms/rest/excel2007-2json.action	GET
CSV (default format)	https://your_confluence_server/ajax/confiforms/rest/csv-2json.action	GET

Services expect/support the following parameters:

Parameter	Description	Type	Required
attachmentId	Attachment ID of the Excel file attachment to use as source	Numeric	Yes
skipRows	Number of rows to skip (from the top). Defaults to 0 (rows to skip, from the top of the sheet)	Numeric	No
idCellNum	Which column to use as ID	Numeric	No
labelCellNum	Which column to use as label	Numeric	No
lookupValue <i>Since 2.27.21</i>	Can supply a lookup value for the service to lookup. The subset returned will be filtered to match (as substring) the given value	Text	No
lookupField <i>Since 2.27.23</i>	Limiting the matching logic to specific field (cell) Example: <i>cell1</i> to limit to <i>cell1</i> value only. If not specified then the match for given lookupValue is done against the whole row (all cells)	Text	No
exactMatch <i>Since 2.27.24</i>	You can set the search to use "exact match", by default (when exactMatch is unspecified or false) the search is done as text substring match (case insensitive)	Boolean	No

 Column and rows numbering starts with 0

 NB! Only the **first** sheet from Excel file is currently supported!

Examples:

https://YOUR_CONFLUENCE_SERVER/ajax/confiforms/rest/excel-2json.action?attachmentId=2195459&skipRows=1&idCellNum=0&labelCellNum=1

https://YOUR_CONFLUENCE_SERVER/ajax/confiforms/rest/excel2007-2json.action?attachmentId=2195460&skipRows=1

Last example reads ALL the rows and columns (skipping just the first one) and creates a structure similar to the one listed below

```
[
  {
    "cell12": "uno",
    "cell13": "in cell 3",
    "cell11": "1"
  },
  {
    "cell12": "duo",
    "cell11": "2"
  },
  {
    "cell12": "tres",
    "cell13": "here is cell three",
    "cell11": "3"
  },
  {
    "cell12": "four",
    "cell11": "4"
  },
  {
    "cell12": "five",
    "cell11": "5"
  },
  {
    "cell12": "six",
    "cell11": "6"
  }
]
```

From the following excel file



Book1.xls

As you can see, we can create a JSON Array structure that perfectly fits our ConfiForms web-service enabled fields

So, here is how you can create a ConfiForms Form which uses this file and shows it's values in the dropdown

Configuration for ConfiForms looks like this

The screenshot shows a nested configuration structure for ConfiForms. At the top is a header bar labeled "ConfiForms Form (Definition) | formName = myform". Below this is a container for "ConfiForms (Form View) Registrations Control | embedded = true". Inside this container is a field definition box labeled "ConfiForms Form Field (Definition) | fieldName = myvalues | fiel...". Each element has a small "CONFI" logo with "Form" and "Definition" sub-labels.

With "myvalues" field configured as web-service dropdown

This is a configuration window titled "Insert 'ConfiForms Form Field (Definition)' Macro". It is divided into two main sections. The left section contains configuration options: a note "alphanumeric (contain only latin symbols and/or numbers, no umlauts!)", a "Field label *" text input with "My excel values", a "Required *" checkbox, a "Field type" dropdown menu with "Webservice dropdown" selected, a "Webservice connection" dropdown menu, a "Configure NEW connection" button, and a "Service URL" text input. Below these is a note: "Your service must return an array of JSON objects in order for this field to work. Or you can use the following example (Example)". The right section is titled "Preview" and is currently empty. At the bottom of the window is a "Select macro" button.

We have defined new connection to use the above mentioned excel to JSON APIs

WS Connection Configuration

Name: *

The name for your configuration, will be visible to users

WS URL

Enter with protocol (http:// or https://) or without if you want to use this as a local proxy (in this case the URL has to start with a /). This URL will be combined together with the service URL you setup. For WS-backed fields in ConfiForms you service must accept GET requests and return JSON as a result

Default request
headers

In HeaderName:HeaderValue format, use new line for each new header/value pair

Username

Leave empty, if the connection requires **no authentication**

Password

When NTLM auth shall be used

Save

Close

And set the mapping to use cell1 as ID column for the dropdown and cell2 as a label

Edit 'ConfiForms Form Field (Definition)' Macro

Root to use

When left empty, the complete JSON document will be taken as source using the mapping below

Field to use as "ID"

cell1

You can navigate using fields in a JSON tree, like: myfield.anotherfield. And when 'Root to use' is given, then you start your navigation from a given root.

Field to use as "Label"

cell2

Sames, as with ID field. You can navigate through the JSON tree using fieldname.subfield.someotherfield to point at the specific value

Sort by label field

☐ By default, the choices are not sorted, unless the ID field is left

Preview

Select macro

Save

Cancel

And you can see the online results right here

My excel values * unoduotresfourfivesix

Save

All the registrations so far (and yes, we can access ANY other field, like cell3 in our views, see below)

My excel values	myvalues.id	myvalues.cell1	myvalues.label	myvalues.cell3
duo	c81e728d9d4c2f636f067f89cc14862c	2	duo	
five	e4da3b7fbbce2345d7772b0674a318d5	5	five	
six	1679091c5a880faf6fb5e6087eb1b2dc	6	six	
four	a87ff679a2f3e71d9181a67b7542122c	4	four	
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
duo	c81e728d9d4c2f636f067f89cc14862c	2	duo	

four	a87ff679a2f3e71d9181a67b7542122c	4	four	
six	1679091c5a880faf6fb5e6087eb1b2dc	6	six	
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
six	1679091c5a880faf6fb5e6087eb1b2dc	6	six	
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
four	a87ff679a2f3e71d9181a67b7542122c	4	four	
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
duo	c81e728d9d4c2f636f067f89cc14862c	2	duo	
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
duo	c81e728d9d4c2f636f067f89cc14862c	2	duo	
duo	c81e728d9d4c2f636f067f89cc14862c	2	duo	
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
tres	eccbc87e4b5ce2fe28308fd9f2a7baf3	3	tres	here is cell three
tres	eccbc87e4b5ce2fe28308fd9f2a7baf3	3	tres	here is cell three
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
uno	c4ca4238a0b923820dcc509a6f75849b	1	uno	in cell 3
tres	eccbc87e4b5ce2fe28308fd9f2a7baf3	3	tres	here is cell three