

# Attributes available for each field

Each form must have some fields :D Dynamic forms allow you to create multiple field types. Each form field has a set of standard configurations. Additionally, fields can contain their own configurations.

These are settings that you can activate in any field.

## size

The size of the field is defined on the basis of blocks, divide the screen into 12 parts and determine the size of the field "filling in these parts".12 will set the field size for the whole screen, 6 will set the field size for half the screen.

Screen sizes are defined by the letters **s,m,l (small, medium, large)**. Setting s12 m6 l3 will set the value 12 for small screens, 6 for medium screens and 3 for large screens.

size: s12 m6

> Application for vacation leave



Your name \*

John Example



Phone number

+48 123 321 123

SEND FORM



> Application for vacation leave



Your name \*

John Example



Phone number

+48 123 321 123

SEND FORM

# type

Field type, here we enter the specific field type you want to use e.g. select, text.


# id

field identifier.

This value must be unique and must not contain spaces!

# tooltip

When you hover over the field, a balloon will pop up, displaying any text or HTML.



Your name \*

John Example

1  
2  
3

Phone number

+48 123 321 123

Example tooltip

SEND FORM

# label

Here you can enter the field title e.g. "Your name".

# validate

Here we can enter a list of any field validations (we can also create our own regex). An example of a validator is "required".

```
//example  
"validate": ["required", "email"]
```

## Available validators

### **after**

**alpha** - The field under validation may only contain alphabetic characters.

**alpha\_dash** - The field under validation may contain alphabetic characters, numbers, dashes or underscores.

**alpha\_num** - The field under validation may contain alphabetic characters or numbers.

**alpha\_spaces** - The field under validation may contain alphabetic characters or spaces.

**before** - The field under validation must have a numeric value bounded by a minimum value and a maximum value.

**between** - The field under validation must have a numeric value bounded by a minimum value and a maximum value.

**confirmed** - The field under validation must have the same value as the confirmation field.

### **credit\_card**

### **date\_between**

### **date\_format**

### **decimal**

**digits** - The field under validation must be numeric and have the specified number of digits.

**dimensions** - The file added to the field under validation must be an image (jpg,svg,jpeg,png,bmp,gif) having the exact specified dimension.

**email** - The field under validation must be a valid email.

**ext** - The file added to the field under validation must have one of the extensions specified.

**image** - The file added to the field under validation must have an image mime type (image/\*).

### **in**

**integer** - The field under validation must be a valid integer value. Doesn't accept exponentiale notation.

### **ip**

**is** - The field under validation must match the given value, uses strict equality.

**is\_not** - The field under validation must not match the given value, uses strict equality.

**length** - The field under validation must have exactly have the specified number of items, only works for iterables.

**max** - The field under validation length may not exceed the specified length.

**max\_value** - The field under validation must be a numeric value and must not be greater than the specified value.

**mimes** - The file type added to the field under validation should have one of the specified mime types.

**min** - The field under validation length should not be less than the specified length.

**min\_value** - The field under validation must be a numeric value and must not be less than the specified value.

## not\_in

**numeric** - The field under validation must only consist of numbers.

**regex** - The field under validation must match the specified regular expression.

**required** - The field under validation must have a non-empty value. By default, all validators pass the validation if they have "empty values" unless they are required. Those empty values are: empty strings, `undefined`, `null`, empty arrays.

**size** - The file size added to the field under validation must not exceed the specified size in kilobytes.

## url

Parameters to attributes should be separated by ":" e.g. **digits:11** - means 11 numbers allowed.

## icon

Field icon. We can set any Material Design compatible icons here.

<https://material.io/resources/icons/?style=baseline>

## hidden

Field display.

A field that is hidden still exists on the form (but is invisible) its values can be sent!

## disabled

Field is disabled.

## otrs\_visible

Here we specify where the field is to be added.

default: article

article

The field will be added to the article.

## dynamicfield

The field will be added where dynamic fields are displayed.

Remember that if the field will have an identifier that matches the dynamic field in OTRS, but will not have this value set - it will not be added as a dynamic field!

## all

The field will be added both as a dynamic field (*if the identifier matches the field in OTRS*), and will be added to the note.

## none

The field will not be added anywhere.

## file

# placeholder

Default value displayed in the field as a placeholder

# value

Default field value. Here we can enter, for example, the default value of a selected item from the list or any other thing we want to complete by default on the form.

This value can be changed by the user while filling in the form - unless the field is disabled or hidden.

# api

API mechanism allows you to edit any attributes of each field (after the data is loaded, it will be automatically refreshed and the view will be re-rendered).

The API mechanism was created so that data from external systems can be entered into the forms. Using api, you can change any field parameter at any time when using the form.

Usually the field "api" is used for normal downloading of data from external systems. For example, if you want to enter a default value when downloading from another system, just enter the address of the endpoint, which will return the data in the appropriate format.

The address will be queried as soon as the form is loaded.

Option "api\_watcher" allows for more advanced actions.

Example:

*The field is hidden by default and if it succeeds in getting data from the API it will be shown and its value will change to "New Value from API!"*

Our API is available at <https://example.intalio.pl>

```
{
  "size": "s12 m12",
  "type": "text",
  "max": 13,
  "id": "ExampleFieldID",
  "label": "Example field Label",
  "icon": "confirmation_number",
  "validate": [
    "required"
  ],
  "hidden": true,
  "value": "",
  "api": "https://example.intalio.pl"
},
```

Response:

```
{
  "hidden": false,
  "value": "New Value form API!"
}
```

This way we can change any parameter of each field.

## api\_watcher

Okay, not always easy is the best. That's why we created "api\_watcher."

This mechanism allows you to listen to changes in any fields and then send this information to any API.

This parameter allows you to complete the current field with data that are dependent on another field.

For example:

Directory of services - you choose "Hardware repair", and in the next step you choose the name of a specific item, e.g. "Computer repair".

To make such a form, the detailed field ("Repairing the computer") must listen for changes in the general field "Hardware repair". If there is any change in the general field - the detail field will ask the API and update its data.

*Configuration:*

```
{
  "size": "s12 m12",
  "type": "text",
  "max": 13,
  "id": "ExampleFieldID",
  "label": "Example field Label",
  "icon": "confirmation_number",
  "validate": [
    "required"
  ],
  "hidden": true,
  "value": "",
  "api_watcher": {
    "delay_api": "",
    "fields": [
      {
        "field_id": "ExampleField_Main",
        "api": "https://example.intalio.pl?DynamicFormField=${value}&lang=${LANG}"
      }
    ]
  }
},
```



As you can see in the example above, the "api\_watcher" mechanism allows you to paste the value received from the listened field anywhere. Add **`${value}`** anywhere in the URL and the data will be pasted there. The currently loaded language will appear in place **`${LANG}`**

## Using **`${value}`**

This is the key where the value from the listening field will appear.

## Using **`${LANG}`**

If your system is to support multiple languages, the API must also do it. To send the currently running form language to the API, you can add this parameter.

## fields

"fields" is an array that can contain any number of fields for which the current field is to listen.

## field\_id

This is where you type the field identifier you want to listen to.

## api

The address to be called after changing the data in the listening field.

## api\_add\_lang

Do you include the variable **`${LANG}`** and insert the currently selected language in its place

## add\_session\_id\_to\_api

If you want to send a user session to the API, you can do so by activating this parameter. If it is activated, the system will automatically attach the sessionId parameter to each request.

If you want to authorize users or have access to user data via session, you must use this parameter.

---

Revision #6

Created 24 May 2020 14:24:05 by editor

Updated 16 February 2023 12:16:46 by editor