Actions

Actions define the behavior of the system in response to user actions: login, action button, selection of an invoice, ...

Actions can be stored in the database or returned directly as dictionaries in e.g. button methods. All actions share two mandatory attributes:

type

the category of the current action, determines which fields may be used and how the action is interpreted

name

short user-readable description of the action, may be displayed in the client's interface

A client can get actions in 4 forms:

False

if any action dialog is currently open, close it

A string

if a [client action](https://www.odoo.com/documentation/10.0/reference/actions.html#reference-actions-client) matches, interpret as a client action's tag, otherwise treat as a number

A number

read the corresponding action record from the database, may be a database identifier or an [external id](https://www.odoo.com/documentation/10.0/glossary.html#term-external-id)

A dictionary

treat as a client action descriptor and execute

**Window Actions (ir.actions.act\_window)**

The most common action type, used to present visualisations of a model through [views](https://www.odoo.com/documentation/10.0/reference/views.html#reference-views): a window action defines a set of view types (and possibly specific views) for a model (and possibly specific record of the model).

Its fields are:

res\_model

model to present views for

views

a list of (view\_id, view\_type) pairs. The second element of each pair is the category of the view (tree, form, graph, ...) and the first is an optional database id (or False). If no id is provided, the client should fetch the default view of the specified type for the requested model (this is automatically done by [fields\_view\_get()](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.models.Model.fields_view_get)). The first type of the list is the default view type and will be open by default when the action is executed. Each view type should be present at most once in the list

res\_id (optional)

if the default view is form, specifies the record to load (otherwise a new record should be created)

search\_view\_id (optional)

(id, name) pair, id is the database identifier of a specific search view to load for the action. Defaults to fetching the default search view for the model

target (optional)

whether the views should be open in the main content area (current), in full screen mode (fullscreen) or in a dialog/popup (new). Use main instead of current to clear the breadcrumbs. Defaults to current.

context (optional)

additional context data to pass to the views

domain (optional)

filtering domain to implicitly add to all view search queries

limit (optional)

number of records to display in lists by default. Defaults to 80 in the web client

auto\_search (optional)

whether a search should be performed immediately after loading the default view. Defaults to True

For instance, to open customers (partner with the customer flag set) with list and form views:

{

"type": "ir.actions.act\_window",

"res\_model": "res.partner",

"views": [[False, "tree"], [False, "form"]],

"domain": [["customer", "=", true]],

}

Or to open the form view of a specific product (obtained separately) in a new dialog:

{

"type": "ir.actions.act\_window",

"res\_model": "product.product",

"views": [[False, "form"]],

"res\_id": a\_product\_id,

"target": "new",

}

In-database window actions have a few different fields which should be ignored by clients, mostly to use in composing the views list:

view\_mode

comma-separated list of view types as a string. All of these types will be present in the generated views list (with at least a False view\_id)

view\_ids

M2M[1](https://www.odoo.com/documentation/10.0/reference/actions.html#notquitem2m) to view objects, defines the initial content of views

view\_id

specific view added to the views list in case its type is part of the view\_mode list and not already filled by one of the views in view\_ids

These are mostly used when defining actions from [Data Files](https://www.odoo.com/documentation/10.0/reference/data.html#reference-data):

<record model="ir.actions.act\_window" id="test\_action">

<field name="name">A Test Action</field>

<field name="res\_model">some.model</field>

<field name="view\_mode">graph</field>

<field name="view\_id" ref="my\_specific\_view"/>

</record>

will use the "my\_specific\_view" view even if that's not the default view for the model.

The server-side composition of the views sequence is the following:

* get each (id, type) from view\_ids (ordered by sequence)
* if view\_id is defined and its type isn't already filled, append its (id, type)
* for each unfilled type in view\_mode, append (False, type)

**URL Actions (ir.actions.act\_url)**

Allow opening a URL (website/web page) via an Odoo action. Can be customized via two fields:

url

the address to open when activating the action

target

opens the address in a new window/page if new, replaces the current content with the page if self. Defaults to new

{

"type": "ir.actions.act\_url",

"url": "http://odoo.com",

"target": "self",

}

will replace the current content section by the Odoo home page.

**Server Actions (ir.actions.server)**

Allow triggering complex server code from any valid action location. Only two fields are relevant to clients:

id

the in-database identifier of the server action to run

context (optional)

context data to use when running the server action

In-database records are significantly richer and can perform a number of specific or generic actions based on their state. Some fields (and corresponding behaviors) are shared between states:

model\_id

Odoo model linked to the action, made available in [evaluation contexts](https://www.odoo.com/documentation/10.0/reference/actions.html#reference-actions-server-context)

condition (optional)

evaluated as Python code using the server action's [evaluation context](https://www.odoo.com/documentation/10.0/reference/actions.html#reference-actions-server-context). If False, prevents the action from running. Default: True

Valid action types (state field) are extensible, the default types are:

**code**

The default and most flexible server action type, executes arbitrary Python code with the action's [evaluation context](https://www.odoo.com/documentation/10.0/reference/actions.html#reference-actions-server-context). Only uses one specific type-specific field:

code

a piece of Python code to execute when the action is called

<record model="ir.actions.server" id="print\_instance">

<field name="name">Res Partner Server Action</field>

<field name="model\_id" ref="model\_res\_partner"/>

<field name="code">

raise Warning(object.name)

</field>

</record>

Note

The code segment can define a variable called action, which will be returned to the client as the next action to execute:

<record model="ir.actions.server" id="print\_instance">

<field name="name">Res Partner Server Action</field>

<field name="model\_id" ref="model\_res\_partner"/>

<field name="code">

if object.some\_condition():

action = {

"type": "ir.actions.act\_window",

"view\_mode": "form",

"res\_model": object.\_name,

"res\_id": object.id,

}

</field>

</record>

will ask the client to open a form for the record if it fulfills some condition

This tends to be the only action type created from [data files](https://www.odoo.com/documentation/10.0/reference/data.html#reference-data), other types aside from [multi](https://www.odoo.com/documentation/10.0/reference/actions.html#reference-actions-server-multi) are simpler than Python code to define from the UI, but not from [data files](https://www.odoo.com/documentation/10.0/reference/data.html#reference-data).

**object\_create**

Creates a new record, from scratch (via [create()](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.models.Model.create)) or by copying an existing record (via [copy()](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.models.Model.copy))

use\_create

the creation policy, one of:

new

creates a record in the model specified by model\_id

new\_other

creates a record in the model specified by crud\_model\_id

copy\_current

copies the record on which the action was invoked

copy\_other

copies an other record, obtained via ref\_object

fields\_lines

fields to override when creating or copying the record. [One2many](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.One2many) with the fields:

col1

ir.model.fields to set in the model implied by use\_create

value

value for the field, interpreted via type

type

If value, the value field is interpreted as a literal value (possibly converted), if equation the value field is interpreted as a Python expression and evaluated

crud\_model\_id

model in which to create a new record, if use\_create is set to new\_other

ref\_object

[Reference](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Reference) to an arbitrary record to copy, used if use\_create is set to copy\_other

link\_new\_record

boolean flag linking the newly created record to the current one via a many2one field specified through link\_field\_id, defaults to False

link\_field\_id

many2one to ir.model.fields, specifies the current record's m2o field on which the newly created record should be set (models should match)

**object\_write**

Similar to [object\_create](https://www.odoo.com/documentation/10.0/reference/actions.html#reference-actions-server-object-create) but alters an existing records instead of creating one

use\_write

write policy, one of:

current

write to the current record

other

write to an other record selected via crud\_model\_id and ref\_object

expression

write to an other record whose model is selected via crud\_model\_id and whose id is selected by evaluating write\_expression

write\_expression

Python expression returning a record or an object id, used when use\_write is set to expression in order to decide which record should be modified

fields\_lines

see [object\_create](https://www.odoo.com/documentation/10.0/reference/actions.html#reference-actions-server-object-create)

crud\_model\_id

see [object\_create](https://www.odoo.com/documentation/10.0/reference/actions.html#reference-actions-server-object-create)

ref\_object

see [object\_create](https://www.odoo.com/documentation/10.0/reference/actions.html#reference-actions-server-object-create)

**multi**

Executes multiple actions one after the other. Actions to execute are defined via the child\_ids m2m. If sub-actions themselves return actions, the last one will be returned to the client as the multi's own next action

**trigger**

Sends a signal to a workflow.

wkf\_transition\_id

[Many2one](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2one) to a workflow.transition to trigger

use\_relational\_model

if base (the default), trigger the signal on behalf of the current record. If relational, trigger the signal on behalf of a field of the current record selected through wkf\_model\_id and wkf\_field\_id

**client\_action**

Indirection for directly returning an other action defined using action\_id. Simply returns that action to the client for execution.

**Evaluation context**

A number of keys are available in the evaluation context of or surrounding server actions:

model

the model object linked to the action via model\_id

object, obj

only available if active\_model and active\_id are provided (via context) otherwise None. The actual record selected by active\_id

pool

the current database registry

datetime, dateutil, time

corresponding Python modules

cr

the current cursor

user

the current user record

context

execution context

Warning

constructor for the Warning exception

**Report Actions (ir.actions.report.xml)**

Triggers the printing of a report

name (mandatory)

only useful as a mnemonic/description of the report when looking for one in a list of some sort

model (mandatory)

the model your report will be about

report\_type (mandatory)

either qweb-pdf for PDF reports or qweb-html for HTML

report\_name

the name of your report (which will be the name of the PDF output)

groups\_id

[Many2many](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2many) field to the groups allowed to view/use the current report

paperformat\_id

[Many2one](https://www.odoo.com/documentation/10.0/reference/orm.html#odoo.fields.Many2one) field to the paper format you wish to use for this report (if not specified, the company format will be used)

attachment\_use

if set to True, the report is only generated once the first time it is requested, and re-printed from the stored report afterwards instead of being re-generated every time.

Can be used for reports which must only be generated once (e.g. for legal reasons)

attachment

python expression that defines the name of the report; the record is accessible as the variable object

**Client Actions (ir.actions.client)**

Triggers an action implemented entirely in the client.

tag

the client-side identifier of the action, an arbitrary string which the client should know how to react to

params (optional)

a Python dictionary of additional data to send to the client, alongside the client action tag

target (optional)

whether the client action should be open in the main content area (current), in full screen mode (fullscreen) or in a dialog/popup (new). Use main instead of current to clear the breadcrumbs. Defaults to current.

{

"type": "ir.actions.client",

"tag": "pos.ui"

}

tells the client to start the Point of Sale interface, the server has no idea how the POS interface works.

[[1]](https://www.odoo.com/documentation/10.0/reference/actions.html#id1) technically not an M2M: adds a sequence field and may be composed of just a view type, without a view id.