Odoo REST API: Version 1.0 Documentation

Release 1.0.1

Synconics Technologies Pvt. Ltd. (https://www.synconics.com)

Jun 17, 2021

Contents

Our Odoo REST API Reference houses a lot of information, but doesn't always tell you how you should use it.

If you want to built apps and other integrations for the Odoo, this tutorial will walk you through what is required to authenticate and make basic API calls.

CHAPTER 1

Get the module

The module **restapi** is available on **Odoo App Store**, Here are links for:

- Version 12.0 (Community & Enterprise)
- Version 13.0 (Community & Enterprise)
- Version 14.0 (Community & Enterprise)

CHAPTER 2

Installation

Install **restapi** module by following below steps:

1. Unzip **restapi** module to **custom addons** directory
2. Restart odoo server
3. Activate **Developer Mode** from the **Settings** menu
4. Navigate to the **Apps** menu
5. Click on **Update Apps List** menu in left side bar
6. Once apps list is updated, click on **Apps** menu from left / top side bar
7. Search module **restapi**
8. Click on **Install** button.

CHAPTER 3

Getting Started

3.1 Connection

3.1.1 Configuration

If you already have an **Odoo server** and **restapi** module installed, you just need to follow below steps

---

Note: To use REST API on Odoo, you will need to set consumer key and secret for OAuth application on the user account you want to use:

- Log in your Odoo instance with an **administrator** account
- Go to **Settings** → **Users** → **Users**
- Click on the **user** you want to use for REST API access
- Click the **OAuth applications** button
- Register an **Application** you want to interact with your Odoo instance
- Click on **Save** button to generate **Consumer Key** and **Secret**.

3.1.2 Demo

To make exploration simpler, you can also ask https://odoo-restapi-demo.synconics.com for a test database:

GET /start

Request:

```
GET /start HTTP/1.1
Host: odoo-restapi-demo.synconics.com
```

Response:

```
HTTP/1.1 200 OK

{
  'client_id': 'uwCrAHAQbL7D9cvJLIztNaZ0bziEGMDh',
  'client_secret': 'FtHzOQVEs0aSEL9AXuIe9k7X6E2MekU7',
  'host': 'odoo-restapi-demo.synconics.com',
```

```
  'database': 'odoo_restapi_demo'
}
```

Request Headers

- Accept – the response content type depends on *Accept* header

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource

3.1.3 Logging In

Odoo requires users of the REST API to be authenticated before they can query most data.

The `restapi/1.0/common` endpoint provides meta-calls which don't require authentication, such as the authentication itself or fetching version information. To verify if the connection information is correct before trying to authenticate, the simplest call is to ask for the server's version through the `restapi/1.0/common/version` endpoint. The authentication itself is done through the **OAuth 1.0** `restapi/1.0/common/oauth1` or **OAuth 2.0** `restapi/1.0/common/oauth2` endpoints.

How you can do

Odoo Version Information

The `restapi/1.0/common/version` endpoint provides Odoo server version information which don't require authentication.

GET /restapi/1.0/common/version

Request:

```
GET /restapi/1.0/common/version HTTP/1.1
Host: <your Odoo server url>
```

Response:

```
HTTP/1.1 200 OK

{
  'server_version': '14.0',
  'server_version_info': [14, 0, 0, "final", 0],
  'server_series': '14.0',
  'protocal_version': 1
}
```

Request Headers

- Accept – the response content type depends on *Accept* header

Response Headers

- Content-Type – this depends on `Accept` header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no user

OAuth1 Authentication

Start with setting up a new consumer by following the instructions on *Configuration*. When you have obtained a `key` and a `secret` you can try out OAuth 1.0 `resapi/1.0/common/oauth1` flow goes as follows to get authorized:

Note: OAuth endpoints:

1. *POST {your_Odoo_server_url}/restapi/1.0/common/oauth1/request_token* (Temporary Credential Request endpoint)
2. *GET {your_Odoo_server_url}/restapi/1.0/common/oauth1/authorize* (Resource Owner Authorization endpoint)
3. *POST {your_Odoo_server_url}/restapi/1.0/common/oauth1/access_token* (Token Credentials Request endpoint)

---

1. Temporary Credential Request

Obtain a request token which will identify you (the consumer) in the next step. At this stage you will only need your consumer key and secret.

`POST /restapi/1.0/common/oauth1/request_token`

Request:

```
POST /restapi/1.0/common/oauth1/request_token HTTP/1.1
Host: {your_Odoo_server_url}
Authorization: OAuth oauth_consumer_key='uwCrAHAQbL7D9cvJLIztNaZ0bziEGMDh',
                     oauth_nonce='71257790252100875101500704380',
                     oauth_callback='https%3A%2F%2F127.0.0.1%2Fcallback',
                     oauth_signature_method='HMAC-SHA1',
                     oauth_timestamp='1500704388',
                     oauth_signature='KbLt0XDVjljXhMJHmmpWxHkFnfs%3D',
                     oauth_version='1.0'
```

Response:

```
HTTP/1.1 200 OK

{
  'oauth_token': 'mXYKtuv8k3NJfnpLMpU3KFuEijXx2Aat',
  'oauth_token_secret': 'QAlvAmzyULWeitpe24oNhj3n91los7W5'
}
```

Query Parameters

- **`oauth_consumer_key`** – Odoo `consumer key`
- **`oauth_nonce`** – A randomly selected value provided by your application, which is unique for each authorization request. During the OAuth callback phase, your application must

check that this value matches the one you provided during authorization. This mechanism is important for the security of your application.

- **oauth_callback** – An absolute URL to which the Odoo will redirect the User back when the Obtaining User Authorization step is completed.
- **oauth_signature_method** – The signature method that Consumer used to sign the request. The protocol defines three signature methods: `HMAC-SHA1`, `RSA-SHA1`, and `PLAINTEXT`.
- **oauth_timestamp** – The timestamp is expressed in the number of seconds since January 1, 1970 00:00:00 GMT. The timestamp value MUST be a positive integer and MUST be equal or greater than the timestamp used in previous requests.
- **oauth_singature** – Base64-encoded HMAC-SHA256 signature signed with the consumer's private key containing the all the components of the request and some OAuth value. The signature can be used to verify that the identity URL wasn't modified because it was sent by the server.
- **oauth_version** – OPTIONAL. If present, value MUST be 1.0. Odoo assume the protocol version to be 1.0 if this parameter is not present. Odoo's response to non-1.0 value is left undefined.

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed

2. Resource Owner Authorization

Obtain authorization from the user (resource owner) to access their protected resources (customers, orders, etc.). This is commonly done by redirecting the user to a specific url to which you add the request token as a query parameter. Note that not all services will give you a verifier even if they should. Also the oauth_token given here will be the same as the one in the previous step.

GET /restapi/1.0/common/oauth1/authorize

Request:

```
GET /restapi/1.0/common/oauth1/authorize HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'oauth_token': 'mXYKtuv8k3NJfnpLMpU3KFuEijXx2Aat',
```

```
  'oauth_verifier': 'sdflk3450FASDLJasd2349dfs'
}
```

Query Parameters

- **oauth_token** – OPTIONAL. The Request Token obtained in the previous step.

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – OPTIONAL OAuth token to authenticate

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed

3. Token Credentials Request

Obtain an access token from the Odoo. Save this token as it can be re-used later. In this step we will re-use most of the credentials obtained uptil this point.

POST /restapi/1.0/common/oauth1/access_token

Request:

```
POST /restapi/1.0/common/oauth1/access_token HTTP/1.1
Host: {your_Odoo_server_url}
Authorization: OAuth oauth_consumer_key='uwCrAHAQbL7D9cvJLIztNaZ0bziEGMDh',
                     oauth_token='mXYKtuv8k3NJfnpLMpU3KFuEijXx2Aat',
                     oauth_nonce='156754554473268986001500738176',
                     oauth_signature_method='HMAC-SHA1',
                     oauth_timestamp='1500738189',
                     oauth_verifier='sdflk3450FASDLJasd2349dfs',
                     oauth_signature='KbLt0XDVjljXhMJHmmpWxHkFnfs%3D',
                     oauth_version='1.0'
```

Response:

```
HTTP/1.1 200 OK

{
  'oauth_token': 'RF7gImCv0B58eogLiPmOmNPizZEVVUWP',
  'oauth_token_secret': 'oxBUTIjTl8gfbxEv2jpXo5rRtQ16u3Lg'
}
```

Query Parameters

- **oauth_consumer_key** – Odoo `consumer key`
- **oauth_token** – The Request Token obtained previously.

- **oauth_nonce** – A randomly selected value provided by your application, which is unique for each authorization request. During the OAuth callback phase, your application must check that this value matches the one you provided during authorization. This mechanism is important for the security of your application.
- **oauth_signature_method** – The signature method that Consumer used to sign the request. The protocol defines three signature methods: `HMAC-SHA1`, `RSA-SHA1`, and `PLAINTEXT`.
- **oauth_timestamp** – The timestamp is expressed in the number of seconds since January 1, 1970 00:00:00 GMT. The timestamp value MUST be a positive integer and MUST be equal or greater than the timestamp used in previous requests.
- **oauth_verifier** – The verification code received from the Odoo.
- **oauth_singature** – Base64-encoded HMAC-SHA256 signature signed with the consumer's private key containing the all the components of the request and some OAuth value. The signature can be used to verify that the identity URL wasn't modified because it was sent by the server.
- **oauth_version** – OPTIONAL. If present, value MUST be 1.0. Odoo assume the protocol version to be 1.0 if this parameter is not present. Odoo's response to non-1.0 value is left undefined.

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed

OAuth2 Authentication

Setup credentials following the instructions on *Configuration*. When you have obtained a `client_id` and a `client_secret` you can try out OAuth 2.0 `resapi/1.0/common/oauth2` flow goes as follows to get authorized:

Note: OAuth endpoints:

1. *GET {your_Odoo_server_url}/restapi/1.0/common/oauth2/authorize* (Resource Owner Authorization endpoint)
2. *POST {your_Odoo_server_url}/restapi/1.0/common/oauth2/access_token* (Token Credentials Request endpoint)

---

1. Resource Owner Authorization

User authorization through redirection. First we will create an authorization url from the base URL given by the Odoo and the credentials previously obtained.

GET /restapi/1.0/common/oauth2/authorize

Request:

```
GET /restapi/1.0/common/oauth2/authorize HTTP/1.1
Host: {your_Odoo_server_url}
Authorization: OAuth client_id='uwCrAHAQbL7D9cvJLIztNaZ0bziEGMDh',
                     state='Y1Ux1iNPvn6KYQK5Lj84WJ9VJrQw1L',
                     redirect_uri='https%3A%2F%2F127.0.0.1%2Fcallback',
                     response_type='code'
```

Response:

```
HTTP/1.1 200 OK

{
  'code': 'dcee1806d2c50d0fb598',
  'state': 'Y1Ux1iNPvn6KYQK5Lj84WJ9VJrQw1L'
}
```

Query Parameters

- **client_id** – Odoo `consumer key`
- **state** – Specifies any additional URL-encoded state data to be returned in the callback URL after approval.
- **redirect_uri** – An absolute URL to which the Odoo will redirect the User back when the obtaining User Authorization step is completed.
- **response_type** – Must be `code` for this authentication flow.

Request Headers

- Accept – the response content type depends on *`Accept`* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *`Accept`* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed

2. Token Credentials Request

Fetch an access token from the Odoo using the authorization code obtained during user authorization.

POST /restapi/1.0/common/oauth2/access_token

Request:

```
POST /restapi/1.0/common/oauth2/access_token HTTP/1.1
Host: {your_Odoo_server_url}
Authorization: OAuth client_id='uwCrAHAQbL7D9cvJLIztNaZ0bziEGMDh',
                     client_secret='FtHzOQVEs0aSEL9AXuIe9k7X6E2MekU7',
                     redirect_uri='https%3A%2F%2F127.0.0.1%2Fcallback',
```

```
                    code='dcee1806d2c50d0fb598'
                    grant_type='authorization_code'
```

Response:

```
HTTP/1.1 200 OK

{
  'access_token': 'eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIn',
  'token_type': 'bearer',
  'access_token_validity': '7/20/2017 12:00:05',
  'refresh_token': 'ZXIiLCJnaXZlbl9uYW1lIjoiRnJhbmsifQ'
}
```

Query Parameters

- **client_id** – Odoo `consumer key`
- **client_secret** – Odoo `consumer secret`
- **redirect_uri** – An absolute URL to which the Odoo will redirect the User back when the obtaining User Authorization step is completed.
- **code** – Authorization code the consumer must use to obtain the access and refresh tokens.
- **grant_type** – Value must be `authorization_code` for this flow.

Request Headers

- Accept – the response content type depends on *`Accept`* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *`Accept`* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed

3.2 Calling Methods

The second endpoint `restapi/1.0/object` is used to call methods of odoo models.

3.2.1 What you can do

The REST API lets you do the following with the Odoo models:

Check Access Rights

For instance to see if we can read the `res.partner` model we can call `check_access_rights` with `operation` passed by position and `raise_exception` passed by keyword (in order to get a true/false result rather than true/error).

GET /restapi/1.0/object/{object_name}/check_access_rights?operation={list_of_operations}

Request:

```
GET /restapi/1.0/object/res.partner/check_access_rights?operation=['read']&raise_
↪exception=True HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'return': true
}
```

Query Parameters

- **operation** – allowed for the user according to the access rights. one of `create`, `write`, `read` or `unlink`.
- **raise_exception** – OPTIONAL. raise an `Error` or return `None`, depending on the value `True` or `False` (default: True)

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

List Records

Records can be listed and filtered via search(). It takes a mandatory domain filter (possibly empty), and returns the database identifiers of all records matching the filter.

GET /restapi/1.0/object/{object_name}/search

Request:

```
GET /restapi/1.0/object/res.partner/search?domain=[('is_company','=',True),(
↪'customer','=',True)] HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': [
```

```
      7, 18, 12, 10, 17, 19, 8, 31, 26, 16, 13, 20, 30, 22, 29, 15, 23, 28, 74
  ]
}
```

Query Parameters

- **domain** – A search domain. Use an empty list to match all records.
- **offset** – OPTIONAL. Number of results to ignore (default: none)
- **limit** – OPTIONAL. Maximum number of records to return (default: all)
- **order** – OPTIONAL. Sort string
- **count** – OPTIONAL. if True, only counts and returns the number of matching records (default: False)

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

Pagination

By default a `restapi/1.0/object/{object_name}/search` will return the ids of all records matching the condition, which may be a huge number. `offset` and `limit` parameters are available to only retrieve a subset of all matched records.

Request:

```
GET /restapi/1.0/object/res.partner/search?domain=[('is_company','=',True),(
↪'customer','=',True)]&offset=10&limit=5 HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': [
      13, 20, 30, 22, 29
  ]
}
```

Count Records

Rather than retrieve a possibly gigantic list of records and count them, search_count() can be used to retrieve only the number of records matching the query. It takes the same domain filter as search() and no other parameter.

Warning: calling `restapi/1.0/object/{object_name}/search` then `restapi/1.0/object/{object_name}/search_count` (or the other way around) may not yield coherent results if other users are using the server: stored data could have changed between the calls

GET /restapi/1.0/object/{object_name}/search_count

Request:

```
GET /restapi/1.0/object/res.partner/search_count?domain=[('is_company','=',True),(
↪'customer','=',True)] HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'count': 19
}
```

Query Parameters

- **domain** – A search domain. Use an empty list to match all records.

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

Read Records

Note: API endpoints:

- *GET {your_Odoo_server_url}/restapi/1.0/object/{object_name}/{id}* (Read Single Record)
- *GET {your_Odoo_server_url}/restapi/1.0/object/{object_name}?ids={comma_separated_ids}* (Read Record Set)

- *GET {your_Odoo_server_url}/restapi/1.0/object/{object_name}/?domain={comma_separated_list_of_args}* (Read Filter Records)

Read Single Record

Record data is accessible via the read(), which takes a single record id and optionally a list of fields to fetch. By default, it will fetch all the fields the current user can read, which tends to be a huge amount.

GET /restapi/1.0/object/{object_name}/{id}

Request:

```
GET /restapi/1.0/object/res.partner/12 HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': {
      'id': 12,
      'name': 'Think Big Systems',
      'street': '89 Lingfield Tower',
      'street2': false,
      'city': 'London',
      'state_id': false,
      'zip': false,
      'country_id': [486, 'United Kingdom'],
      'create_date': '2017-07-10 11:02:57',
      'create_uid': [1, 'Administrator'],
      'write_date': '2017-07-11 15:08:45',
      'write_uid': [1, 'Administrator'],
      ...
      ...
      ...
  }
}
```

Query Parameters

- **fields** – OPTIONAL. list of field names to return (default is all fields).

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed

- 403 Forbidden – if any error raise

Conversely, picking only three fields deemed interesting.

Request:

```
GET /restapi/1.0/object/res.partner/12?fields=['name','country_id'] HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': {
      'id': 12,
      'name': 'Think Big Systems',
      'country_id': [486, 'United Kingdom']
  }
}
```

Note: even if the `id` field is not requested, it is always returned

Read List Records

Record data is accessible via the read(), which takes a list of ids (as returned by `/restapi/1.0/object/{object_name}/search`) and optionally domain filter and a list of fields to fetch. By default, it will fetch all the fields the current user can read, which tends to be a huge amount.

`GET /restapi/1.0/object/{object_name}?ids={comma_separated_ids}`

Request:

```
GET /restapi/1.0/object/res.partner?ids=12,17 HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': [
      {
          'id': 12,
          'name': 'Think Big Systems',
          'street': '89 Lingfield Tower',
          'street2': false,
          'city': 'London',
          'state_id': false,
          'zip': false,
          'country_id': [486, 'United Kingdom'],
          'create_date': '2017-07-10 11:02:57',
          'create_uid': [1, 'Administrator'],
          'write_date': '2017-07-11 15:08:45',
          'write_uid': [1, 'Administrator'],
          ...
```

```
            ...
            ...
        },
        {
            'id': 17,
            'name': 'Edward Foster',
            'street': '69 rue de Namur',
            'street2': false,
            'city': 'Wavre',
            'state_id': false,
            'zip': '1300',
            'country_id': [274, 'Belgium'],
            'create_date': '2017-07-04 18:10:31',
            'create_uid': [1, 'Administrator'],
            'write_date': '2017-07-04 19:02:59',
            'write_uid': [1, 'Administrator'],
            ...
            ...
            ...
        }
    ]
}
```

Query Parameters

- **`fields`** – OPTIONAL. list of field names to return (default is all fields).

Request Headers

- Accept – the response content type depends on *`Accept`* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *`Accept`* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

Conversely, picking only three fields deemed interesting.

Request:

```
GET /restapi/1.0/object/res.partner?ids=12,17&fields=['name','country_id']␣
↪HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': [
```

```
      {
          'id': 12,
          'name': 'Think Big Systems',
          'country_id': [486, 'United Kingdom']
      },
      {
          'id': 17,
          'name': 'Edward Foster',
          'country_id': [274, 'Belgium']
      }
  ]
}
```

Note: even if the `id` field is not requested, it is always returned

Read Filter Records

Record data is accessible via the `search_read()` (shortcut which as its name suggests is equivalent to a search() followed by a read(), but avoids having to perform two requests and keep ids around).

It takes similar arguments of search() and optionally a list of fields to fetch. By default, it will fetch all the records and relavent fields the current user can read, which tends to be a huge amount.

`GET /restapi/1.0/object/{object_name}/?domain={comma_separated_list_of_args}`

Request:

```
GET /restapi/1.0/object/res.partner?domain=[('is_company','=',True),('customer','=
↪',True)] HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': [
      {
          'id': 12,
          'name': 'Think Big Systems',
          'street': '89 Lingfield Tower',
          'street2': false,
          'city': 'London',
          'state_id': false,
          'zip': false,
          'country_id': [486, 'United Kingdom'],
          'create_date': '2017-07-10 11:02:57',
          'create_uid': [1, 'Administrator'],
          'write_date': '2017-07-11 15:08:45',
          'write_uid': [1, 'Administrator'],
          ...
          ...
          ...
      },
      {
          'id': 17,
```

```
            'name': 'Edward Foster',
            'street': '69 rue de Namur',
            'street2': false,
            'city': 'Wavre',
            'state_id': false,
            'zip': '1300',
            'country_id': [274, 'Belgium'],
            'create_date': '2017-07-04 18:10:31',
            'create_uid': [1, 'Administrator'],
            'write_date': '2017-07-04 19:02:59',
            'write_uid': [1, 'Administrator'],
            ...
            ...
            ...
        },
        ...
        ...
        ...
    ]
}
```

Query Parameters

- **domain** – OPTIONAL. A search domain. Use an empty list to match all records.
- **fields** – OPTIONAL. list of field names to return (default is all fields).
- **offset** – OPTIONAL. Number of results to ignore (default: none)
- **limit** – OPTIONAL. Maximum number of records to return (default: all)
- **order** – OPTIONAL. Sort string
- **count** – OPTIONAL. if True, only counts and returns the number of matching records (default: False)

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

Conversely, picking only three fields deemed interesting.

Request:

```
GET /restapi/1.0/object/res.partner?domain=[('is_company','=',True),(
↪'customer','=',True)]&fields=['name','country_id']&limit=5 HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': [
      {
          'id': 7,
          'name': 'Agrolait',
          'country_id': [274, 'Belgium']
      },
      {
          'id': 12,
          'name': 'Think Big Systems',
          'country_id': [486, 'United Kingdom']
      },
      {
          'id': 17,
          'name': 'Edward Foster',
          'country_id': [274, 'Belgium']
      },
      {
          'id': 8,
          'name': 'China Export',
          'country_id': [302, 'China']
      },
      {
          'id': 10,
          'name': 'The Jackson Group',
          'country_id': [488, 'United States']
      }
  ]
}
```

Note: even if the `id` field is not requested, it is always returned

Listing Record Fields

fields_get() can be used to inspect a model's fields and check which ones seem to be of interest.

Because it returns a large amount of meta-information (it is also used by client programs) it should be filtered before printing, the most interesting items for a human user are `string` (the field's label), `help` (a help text if available) and `type` (to know which values to expect, or to send when updating a record).

`GET /restapi/1.0/object/{object_name}/fields_get`

Request:

```
GET /restapi/1.0/object/res.partner/fields_get?allfields=[]&attributes=['string',
↪'help','type'] HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
```

```
  'fields': {
      'ean13': {
          'type': 'char',
          'help': 'BarCode',
          'string': 'EAN13'
      },
      'property_account_position_id': {
          'type': 'many2one',
          'help': 'The fiscal position will determine taxes and accounts used for␣
↪the partner.',
          'string': 'Fiscal Position'
      },
      'signup_valid': {
          'type': 'boolean',
          'help': '',
          'string': 'Signup Token is Valid'
      },
      'date_localization': {
          'type": 'date',
          'help": '',
          'string': 'Geo Localization Date'
      },
      'ref_company_ids': {
          'type': 'one2many',
          'help': '',
          'string': 'Companies that refers to partner'
      },
      'sale_order_count': {
          'type': 'integer',
          'help': '',
          'string': '# of Sales Order'
      },
      'purchase_order_count': {
          'type': 'integer',
          'help': '',
          'string': '# of Purchase Order'
      }
  }
}
```

Query Parameters

- **allfields** – OPTIONAL. list of fields to document, all if empty or not provided
- **attributes** – OPTIONAL. list of description attributes to return for each field, all if empty or not provided

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error

- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

Create Records

Records of a model are created using create().

It takes a mapping of fields to values, used to initialize the record. For any field which has a default value and is not set through the mapping argument, the default value will be used.

Warning: while most value types are what would be expected (integer for Integer, string for Char or Text),

- Date, Datetime and `Binary` fields use string values
- One2many and Many2many use a special command protocol detailed in the documentation to the write method.

`POST /restapi/1.0/object/{object_name}?vals={values_for_the_object's_fields}`

Request:

```
POST /restapi/1.0/object/res.partner?vals={'name':'Peter Mitchell','street':'31␣
↪Hong Kong street','city':'Taipei','zip':'106','country_id':482} HTTP/1.1
Host: <your Odoo server url>
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': {
      'id': 20,
      'name': 'Peter Mitchell',
      'street': '31 Hong Kong street',
      'street2': false,
      'city': 'Taipei',
      'state_id': false,
      'zip': '106',
      'country_id': [482, 'Taiwan'],
      'create_date': '2017-07-12 13:34:22',
      'create_uid': [1, 'Administrator'],
      'write_date': '2017-07-12 13:34:22',
      'write_uid': [1, 'Administrator'],
      ...
      ...
      ...
  }
}
```

Query Parameters

- **`vals`** – values for the object's fields, as a dictionary:: `{'field_name': field_value, ...}` see write() for details.

Request Headers

- Accept – the response content type depends on `Accept` header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on `Accept` header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

Update Records

Note: API endpoints:

- *PUT {your_Odoo_server_url}/restapi/1.0/object/{object_name}/{id}?vals={fields_and_values_to_update}* (Update Single Record)
- *PUT {your_Odoo_server_url}/restapi/1.0/object/{object_name}?ids={comma_separated_ids}&vals={fields_and_values_to_upd* (Update Record Set)

Update Single Record

Record can be updated using write(), it takes a record id to update and a mapping of updated fields to values similar to create().

PUT /restapi/1.0/object/{object_name}/{id}?vals={fields_and_values_to_update}

Request:

```
PUT /restapi/1.0/object/res.partner/20?vals={'street2':'Chung Hsiao East Road'}␣
↪HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': {
      'id': 20,
      'name': 'Peter Mitchell',
      'street': '31 Hong Kong street',
      'street2': 'Chung Hsiao East Road',
      'city': 'Taipei',
      'state_id': false,
      'zip': '106',
      'country_id': [482, 'Taiwan'],
      'create_date': '2017-07-12 13:34:22',
      'create_uid': [1, 'Administrator'],
      'write_date': '2017-07-13 11:18:28',
```

```
      'write_uid': [1, 'Administrator'],
      ...
      ...
      ...
  }
}
```

Query Parameters

- **vals** – fields to update and the value to set on them:: `{'field_name': field_value, ...}` see write() for details.

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

Update List Records

Records can be updated using write(), it takes a list of records to update and a mapping of updated fields to values similar to create().

Multiple records can be updated simultanously, but they will all get the same values for the fields being set. It is not currently possible to perform **computed** updates (where the value being set depends on an existing value of a record).

PUT /restapi/1.0/object/{object_name}?ids={comma_separated_ids}&vals={fields_and_values_to_

Request:

```
PUT /restapi/1.0/object/res.partner?ids=17,20&vals={'street2':'Chung Hsiao East␣
↪Road'} HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Partner': [
      {
          'id': 17,
          'name': 'Edward Foster',
          'street': '69 rue de Namur',
          'street2': 'Chung Hsiao East Road',
          'city': 'Wavre',
          'state_id': false,
```

```
            'zip': '1300',
            'country_id': [274, 'Belgium'],
            'create_date': '2017-07-04 18:10:31',
            'create_uid': [1, 'Administrator'],
            'write_date': '2017-07-13 11:18:28',
            'write_uid': [1, 'Administrator'],
            ...
            ...
            ...
        },
        {
            'id': 20,
            'name': 'Peter Mitchell',
            'street': '31 Hong Kong street',
            'street2': 'Chung Hsiao East Road',
            'city': 'Taipei',
            'state_id': false,
            'zip': '106',
            'country_id': [482, 'Taiwan'],
            'create_date': '2017-07-12 13:34:22',
            'create_uid': [1, 'Administrator'],
            'write_date': '2017-07-13 11:18:28',
            'write_uid': [1, 'Administrator'],
            ...
            ...
            ...
        }
    ]
}
```

Query Parameters

- **vals** – fields to update and the value to set on them:: `{'field_name': field_value, ...}` see write() for details.

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

Delete Records

Note: API endpoints:

- *DELETE {your_Odoo_server_url}/restapi/1.0/object/{object_name}/{id}* (Delete Single Record)

- *DELETE {your_Odoo_server_url}/restapi/1.0/object/{object_name}?ids={comma_separated_ids}* (Delete Record Set)

Delete Single Record

Record can be deleted using unlink(), it takes a record id to delete.

DELETE /restapi/1.0/object/{object_name}/{id}

Request:

```
DELETE /restapi/1.0/object/res.partner/20 HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{}
```

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

Delete List Records

Records can be deleted using unlink(), it takes a list of records to delete.

DELETE /restapi/1.0/object/{object_name}?ids={comma_separated_ids}

Request:

```
DELETE /restapi/1.0/object/res.partner?ids=17,20 HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{}
```

Request Headers

- Accept – the response content type depends on *Accept* header

- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

3.3 Report Printing

Available reports can be listed by searching the `ir.actions.report` model, fields of interest being

model the model on which the report applies, can be used to look for available reports on a specific model

name human-readable report name

report_name the technical name of the report, used to print it

Reports can be printed using `restapi/1.0/report` endpoint over REST API with the following information:

- the name of the report (`report_name`)
- the single record id or ids of the records to include in the report

---

Note: API endpoints:

- *GET {your_Odoo_server_url}/restapi/1.0/report/{report_name}/{id}* (Print Single Report)
- *GET {your_Odoo_server_url}/restapi/1.0/report/{report_name}?ids={comma_separated_ids}* (Print Report Set)

3.3.1 Print Single Report

Report can be printed by giving the name of the report(`report_name`) and a single record id.

GET /restapi/1.0/report/{report_name}/{id}

Request:

> **Warning:** this example needs `account` module installed

```
GET /restapi/1.0/report/account.report_invoice/12 HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

Note: the report is sent as PDF binary data encoded in base64, it must be decoded and may need to be saved to disk before use

---

```
HTTP/1.1 200 OK

{
  'report': {
      'id': 12,
      'state': 'true',
      'format': 'pdf',
      'result': 'R0lGODlhbgCMAPf\/APbr48VySrxTO7IgKt2qmKQdJeK8lsFjROG5p\
      /nz7Zg3\nMNmnd7Q1MLNVS9GId71hSJMZIuzTu4UtKbeEeakhKMl8U8WYjfr18YQaIbAf\n
      KKwhKdKzqpQtLebFortOOejKrOjZ1Mt7aMNpVbAqLLV7bsNqR+3WwMqEWenN\n
      sZYxL\/Ddy\/Pm2e7ZxLlUQrIjNPXp3bU5MbhENbEtLtqhj5ZQTfHh0bMxL7Ip\n
      NsNyUYkZIrZJPcqGdYIUHb5aPKkeJnoUHd2yiJkiLKYiKLRFOsyJXKVDO8up\n
      osFaS+TBnK4kKti5sNaYg\/z49aqYl5kqLrljUtORfMOlo\/36+H4ZH
      ...
      ...
      ...'
  }
}
```

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

3.3.2 Print List Reports

Report can be printed by giving the name of the report(`report_name`) and a list of records.

GET /restapi/1.0/report/{report_name}?ids={comma_separated_ids}

Request:

> **Warning:** this example needs `account` module installed

```
GET /restapi/1.0/report/account.report_invoice?ids=12,17 HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

Note: the report is sent as PDF binary data encoded in base64, it must be decoded and may need to be saved to disk before use

---

```
HTTP/1.1 200 OK

{
  'report': [
      {
          'id': 12,
          'state': 'true',
          'format': 'pdf',
          'result': 'R0lGODlhbgCMAPf\/APbr48VySrxTO7IgKt2qmKQdJeK8lsFjROG5p\
          /nz7Zg3\nMNmnd7Q1MLNVS9GId71hSJMZIuzTu4UtKbeEeakhKMl8U8WYjfr18YQaIbAf\n
          KKwhKdKzqpQtLebFortOOejKrOjZ1Mt7aMNpVbAqLLV7bsNqR+3WwMqEWenN\n
          sZYxL\/Ddy\/Pm2e7ZxLlUQrIjNPXp3bU5MbhENbEtLtqhj5ZQTfHh0bMxL7Ip\n
          NsNyUYkZIrZJPcqGdYIUHb5aPKkeJnoUHd2yiJkiLKYiKLRFOsyJXKVDO8up\n
          osFaS+TBnK4kKti5sNaYg\/z49aqYl5kqLrljUtORfMOlo\/36+H4ZH
          ...
          ...
          ...'
      },
      {
          'id': 17,
          'state': 'true',
          'format': 'pdf',
          'result': '9iggMd1TLtbRKUdKXEQFsd4XrZRPLIgMZUeJ+jKvrAlK6AhJ65A\nMp
          MpKuC3j5obIsRwS7hAN8l\/YtvDvnYXHbAoLI47SIUsOMenorF4gO\/
↪m4+fH\npo4vLZ8oKMukqp0cJbhVSMV2U
          uPR0bAfMLIrLrg\/OcJwT8h+Vt+wn8eurLlh\nQrIfKHQOHHQOHf\/\/\/\/\/
          \/\/yH5BAEAAP8ALAAAAABuAIwAAAj\/AP8JHDhQXjpz\n\/
↪PopXNiPn0OHDRMmbKhQIsOJFS1SxAhxI
          8SHFzVeDBnx48iNBAeeOkcxokeX\nFRdOnAlSokaaLXNujJkxo8iYHRkKtWkzZ
          SsaOXkAWsoUECynsHgoqEW1q
          ...
          ...
          ...'
      }
  ]
}
```

Request Headers

- Accept – the response content type depends on *Accept* header
- Authorization – The OAuth protocol parameters to authenticate.

Response Headers

- Content-Type – this depends on *Accept* header of request

Status Codes

- 200 OK – no error
- 404 Not Found – there's no resource
- 401 Unauthorized – authentication failed
- 403 Forbidden – if any error raise

3.4 Inspection and Introspection

While we previously used fields_get() to query a model and have been using an arbitrary model from the start, Odoo stores most model metadata inside a few meta-models which allow both querying the system and altering models and fields (with some limitations) on the fly over REST API.

Note:

1. *Provides information about Odoo models via its various fields* (ir.model)
2. *Provides information about the fields of Odoo models and allows adding custom fields without using Python code* (ir.model.fields)

3.4.1 ir.model

Provides information about Odoo models via its various fields

name a human-readable description of the model

model the name of each model in the system

state whether the model was generated in Python code (`base`) or by creating an `ir.model` record (`manual`)

field_id list of the model's fields through a One2many to *ir.model.fields*

view_ids One2many to the Views defined for the model

access_ids One2many relation to the Access Control set on the model

---

Note: `ir.model` can be used to:

- query the system for installed models (as a precondition to operations on the model or to explore the system's content)
- get information about a specific model (generally by listing the fields associated with it)
- create new models dynamically over REST API

Warning:

- **custom** model names must start with `x_`
- the `state` must be provided and `manual`, otherwise the model will not be loaded
- it is not possible to add new *methods* to a custom model, only fields

Example

1. Create `x_custom_model` model record in `ir.model` object using *Create Records* API endpoint.

   Request:

```
POST /restapi/1.0/object/ir.model?vals={'name':'Custom Model','model':'x_custom_
↪model','state':'manual'} HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

```
HTTP/1.1 200 OK

{
  'Models': {
      'id': 104,
      'name': 'Custom Model',
      'model': 'x_custom_model',
      'state': 'manual'
      ...
      ...
      ...
  }
}
```

2. Inspect a model `x_custom_model`'s fields using *Listing Record Fields* API endpoint.

Request:

```
GET /restapi/1.0/object/x_custom_model/fields_get?attributes=['string','help
↪','type'] HTTP/1.1
Host: {your_Odoo_server_url}
```

Response:

Note: a custom model will initially contain only the "built-in" fields available on all models

---

```
HTTP/1.1 200 OK

{
  'fields': {
      'create_uid': {
          'type': 'many2one',
          'string': 'Created by'
      },
      'create_date': {
          'type": 'datetime',
          'string': 'Created on'
      },
      '__last_update': {
          'type': 'datetime',
          'string': 'Last Modified on'
      },
      'write_uid': {
          'type': 'many2one',
          'string': 'Last Updated by'
      },
      'write_date': {
          'type': 'datetime',
          'string': 'Last Updated on'
      },
      'display_name': {
          'type': 'char',
          'string': 'Display Name'
      },
      'id": {
```

```
            'type': 'integer',
            'string': 'Id'
        }
    }
}
```

3.4.2 ir.model.fields

Provides information about the fields of Odoo models and allows adding custom fields without using Python code

model_id Many2one to *ir.model* to which the field belongs

name the field's technical name (used in `read` or `write`)

field_description the field's user-readable label (e.g. `string` in `fields_get`)

ttype the type of field to create

state whether the field was created via Python code (`base`) or via `ir.model.fields` (`manual`)

required, readonly, translate enables the corresponding flag on the field

groups field-level access control, a Many2many to `res.groups`

selection, size, on_delete, relation, relation_field, domain type-specific properties and customizations, see the fields documentation for details

---

Note: Like custom models, only new fields created with `state="manual"` are activated as actual fields on the model.

Warning: computed fields can not be added via `ir.model.fields`, some field meta-information (defaults, onchange) can not be set either

Example

1. Create `x_custom` model record in `ir.model` object using *Create Records* API endpoint.

   Request:

```
POST /restapi/1.0/object/ir.model?vals={'name':'Custom Model','model':'x_custom',
↪'state':'manual'} HTTP/1.1
Host: {your_Odoo_server_url}
```

   Response:

```
HTTP/1.1 200 OK

{
  'Models': {
      'id': 105,
      'name': 'Custom Model',
      'model': 'x_custom',
      'state': 'manual'
      ...
```

```
        ...
        ...
    }
}
```

2. Create `x_name` field record in `ir.model.fields` object using *Create Records* API endpoint.

   Request:

```
POST /restapi/1.0/object/ir.model.fields?vals={'model_id':105,'name':'x_name',
↪'ttype':'char','state':'manual','required':True} HTTP/1.1
Host: {your_Odoo_server_url}
```

   Response:

```
HTTP/1.1 200 OK

{
    'Fields': {
        'id': 210,
        'name': 'x_name',
        'model_id': [105, 'Custom Model'],
        'ttype': 'char',
        'state': 'manual',
        'required': True
        ...
        ...
        ...
    }
}
```

3. Create `test record` record in `x_custom` object using *Create Records* API endpoint.

   Request:

```
POST /restapi/1.0/object/x_custom?vals={'x_name':'test record'} HTTP/1.1
Host: {your_Odoo_server_url}
```

   Response:

```
HTTP/1.1 200 OK

{
    'Custom Model': {
        'id': 115,
        'x_name': 'test record',
        'display_name': 'test record',
        'create_date': '2017-07-15 14:31:17',
        'create_uid': [1, 'Administrator'],
        'write_date': '2017-07-15 14:31:17',
        'write_uid': [1, 'Administrator'],
        ...
        ...
        ...
    }
}
```

/restapi /start