Upgrade API

**Introduction**

This document describes the API used to upgrade an Odoo database to a higher version.

It allows a database to be upgraded without ressorting to the html form at <https://upgrade.odoo.com> Although the database will follow the same process described on that form.

The required steps are:

* [creating a request](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-create-method)
* [uploading a database dump](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-upload-method)
* [running the upgrade process](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-process-method)
* [obtaining the status of the database request](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-status-method)
* [downloading the upgraded database dump](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-download-method)

**The methods**

**Creating a database upgrade request**

This action creates a database request with the following information:

* your contract reference
* your email address
* the target version (the Odoo version you want to upgrade to)
* the purpose of your request (test or production)
* the database dump name (required but purely informative)
* optionally the server timezone (for Odoo source version < 6.1)

**The create method**

**https://upgrade.odoo.com/database/v1/create**

Creates a database upgrade request

Parameters

* **contract** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your enterprise contract reference
* **email** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your email address
* **target** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) the Odoo version you want to upgrade to. Valid choices: 6.0, 6.1, 7.0, 8.0
* **aim** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) the purpose of your upgrade database request. Valid choices: test, production.
* **filename** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) a purely informative name for you database dump file
* **timezone** ([str](https://docs.python.org/2/library/functions.html#str)) -- (optional) the timezone used by your server. Only for Odoo source version < 6.1

Returns

request result

Return type

JSON dictionary

The *create* method returns a JSON dictionary containing the following keys:

**failures**

The list of errors.

A list of dictionaries, each dictionary giving information about one particular error. Each dictionary can contain various keys depending of the type of error but you will always get the reason and the message keys:

* reason: the error type
* message: a human friendly message

Some possible keys:

* code: a faulty value
* value: a faulty value
* expected: a list of valid values

See a sample output aside.

* JSON

{

"failures": [

{

"expected": [

"6.0",

"6.1",

"7.0",

"8.0",

],

"message": "Invalid value \"5.0\"",

"reason": "TARGET:INVALID",

"value": "5.0"

},

{

"code": "M123456-abcxyz",

"message": "Can not find contract M123456-abcxyz",

"reason": "CONTRACT:NOT\_FOUND"

}

]

}

**request**

If the *create* method is successful, the value associated to the *request* key will be a dictionary containing various information about the created request:

The most important keys are:

* id: the request id
* key: your private key for this request

These 2 values will be requested by the other methods (upload, process and status)

The other keys will be explained in the section describing the [status method](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-status-method).

**Sample script**

Here are 2 examples of database upgrade request creation using:

* one in the python programming language using the pycurl library
* one in the bash programming language using [curl](http://curl.haxx.se) (tool for transfering data using http) and [jq](https://stedolan.github.io/jq) (JSON processor):
* Python
* Bash

from urllib import urlencode

from io import BytesIO

import pycurl

import json

CREATE\_URL = "https://upgrade.odoo.com/database/v1/create"

CONTRACT = "M123456-abcdef"

AIM = "test"

TARGET = "8.0"

EMAIL = "john.doe@example.com"

FILENAME = "db\_name.dump"

fields = dict([

('aim', AIM),

('email', EMAIL),

('filename', DB\_SOURCE),

('contract', CONTRACT),

('target', TARGET),

])

postfields = urlencode(fields)

c = pycurl.Curl()

c.setopt(pycurl.URL, CREATE\_URL)

c.setopt(c.POSTFIELDS, postfields)

data = BytesIO()

c.setopt(c.WRITEFUNCTION, data.write)

c.perform()

# transform output into a dict:

response = json.loads(data.getvalue())

# get http status:

http\_code = c.getinfo(pycurl.HTTP\_CODE)

c.close()

**Uploading your database dump**

There are 2 methods to upload your database dump:

* the upload method using the HTTPS protocol
* the request\_sftp\_access method using the SFTP protocol

**The upload method**

It's the most simple and most straightforward way of uploading your database dump. It uses the HTTPS protocol.

**https://upgrade.odoo.com/database/v1/upload**

Uploads a database dump

Parameters

* **key** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your private key
* **request** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your request id

Returns

request result

Return type

JSON dictionary

The request id and the private key are obtained using the [create method](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-create-method)

The result is a JSON dictionary containing the list of failures, which should be empty if everything went fine.

* Python
* Bash

import os

import pycurl

from urllib import urlencode

UPLOAD\_URL = "https://upgrade.odoo.com/database/v1/upload"

DUMPFILE = "openchs.70.cdump"

fields = dict([

('request', '10534'),

('key', 'Aw7pItGVKFuZ\_FOR3U8VFQ=='),

])

headers = {"Content-Type": "application/octet-stream"}

postfields = urlencode(fields)

c = pycurl.Curl()

c.setopt(pycurl.URL, UPLOAD\_URL+"?"+postfields)

c.setopt(pycurl.POST, 1)

filesize = os.path.getsize(DUMPFILE)

c.setopt(pycurl.POSTFIELDSIZE, filesize)

fp = open(DUMPFILE, 'rb')

c.setopt(pycurl.READFUNCTION, fp.read)

c.setopt(

pycurl.HTTPHEADER,

['%s: %s' % (k, headers[k]) for k in headers])

c.perform()

c.close()

**The request\_sftp\_access method**

This method is recommanded for big database dumps. It uses the SFTP protocol and supports resuming.

It will create a temporary SFTP server where you can connect to and allow you to upload your database dump using an SFTP client.

**https://upgrade.odoo.com/database/v1/request\_sftp\_access**

Creates an SFTP server

Parameters

* **key** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your private key
* **request** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your request id
* **ssh\_keys** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) the path to a file listing the ssh public keys you'd like to use

Returns

request result

Return type

JSON dictionary

The request id and the private key are obtained using the [create method](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-create-method)

The file listing your ssh public keys should be roughly similar to a standard authorized\_keys file. This file should only contains public keys, blank lines or comments (lines starting with the # character)

Your database upgrade request should be in the draft state.

The request\_sftp\_access method returns a JSON dictionary containing the following keys:

* Python
* Bash

import os

import pycurl

from urllib import urlencode

UPLOAD\_URL = "https://upgrade.odoo.com/database/v1/request\_sftp\_access"

SSH\_KEYS="/path/to/your/authorized\_keys"

fields = dict([

('request', '10534'),

('key', 'Aw7pItGVKFuZ\_FOR3U8VFQ=='),

])

postfields = urlencode(fields)

c = pycurl.Curl()

c.setopt(pycurl.URL, UPLOAD\_URL+"?"+postfields)

c.setopt(pycurl.POST, 1)

c.setopt(c.HTTPPOST,[("ssh\_keys",

(c.FORM\_FILE, SSH\_KEYS,

c.FORM\_CONTENTTYPE, "text/plain"))

])

c.perform()

c.close()

**failures**

The list of errors. See [failures](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-json-failure) for an explanation about the JSON dictionary returned in case of failure.

**request**

If the call is successful, the value associated to the *request* key will be a dictionary containing your SFTP connexion parameters:

* hostname: the host address to connect to
* sftp\_port: the port to connect to
* sftp\_user: the SFTP user to use for connecting
* shared\_file: the filename you need to use (identical to the filename value you have used when creating the request in the [create method](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-create-method).)
* request\_id: the related upgrade request id (only informative, ,not required for the connection)
* sample\_command: a sample command using the 'sftp' client

You should normally be able to connect using the sample command as is.

You will only have access to the shared\_file. No other files will be accessible and you will not be able to create new files in your shared environment on the SFTP server.

**Using the 'sftp' client**

Once you have successfully connected using your SFTP client, you can upload your database dump. Here is a sample session using the 'sftp' client:

$ sftp -P 2200 user\_10534@upgrade.odoo.com

Connected to upgrade.odoo.com.

sftp> put /path/to/openchs.70.cdump openchs.70.cdump

Uploading /path/to/openchs.70.cdump to /openchs.70.cdump

sftp> ls -l openchs.70.cdump

-rw-rw-rw- 0 0 0 849920 Aug 30 15:58 openchs.70.cdump

If your connection is interrupted, you can continue your file transfer using the -a command line switch:

sftp> put -a /path/to/openchs.70.cdump openchs.70.cdump

Resuming upload of /path/to/openchs.70.cdump to /openchs.70.cdump

If you don't want to manually type the command and need to automate your database upgrade using a script, you can use a batch file or pipe your commands to 'sftp':

echo "put /path/to/openchs.70.cdump openchs.70.cdump" | sftp -b - -P 2200 user\_10534@upgrade.odoo.com

The -b parameter takes a filename. If the filename is -, it reads the commands from standard input.

**Asking to process your request**

This action ask the Upgrade Platform to process your database dump.

**The process method**

**https://upgrade.odoo.com/database/v1/process**

Process a database dump

Parameters

* **key** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your private key
* **request** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your request id

Returns

request result

Return type

JSON dictionary

The request id and the private key are obtained using the [create method](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-create-method)

The result is a JSON dictionary containing the list of failures, which should be empty if everything went fine.

* Python
* Bash

from urllib import urlencode

from io import BytesIO

import pycurl

import json

PROCESS\_URL = "https://upgrade.odoo.com/database/v1/process"

fields = dict([

('request', '10534'),

('key', 'Aw7pItGVKFuZ\_FOR3U8VFQ=='),

])

postfields = urlencode(fields)

c = pycurl.Curl()

c.setopt(pycurl.URL, PROCESS\_URL)

c.setopt(c.POSTFIELDS, postfields)

data = BytesIO()

c.setopt(c.WRITEFUNCTION, data.write)

c.perform()

# transform output into a dict:

response = json.loads(data.getvalue())

# get http status:

http\_code = c.getinfo(pycurl.HTTP\_CODE)

c.close()

**Obtaining your request status**

This action ask the status of your database upgrade request.

**The status method**

**https://upgrade.odoo.com/database/v1/status**

Ask the status of a database upgrade request

Parameters

* **key** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your private key
* **request** ([str](https://docs.python.org/2/library/functions.html#str)) -- (required) your request id

Returns

request result

Return type

JSON dictionary

The request id and the private key are obtained using the [create method](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-create-method)

The result is a JSON dictionary containing various information about your database upgrade request.

* Python
* Bash

from urllib import urlencode

from io import BytesIO

import pycurl

import json

STATUS\_URL = "https://upgrade.odoo.com/database/v1/status"

fields = dict([

('request', '10534'),

('key', 'Aw7pItGVKFuZ\_FOR3U8VFQ=='),

])

postfields = urlencode(fields)

c = pycurl.Curl()

c.setopt(pycurl.URL, PROCESS\_URL)

c.setopt(c.POSTFIELDS, postfields)

data = BytesIO()

c.setopt(c.WRITEFUNCTION, data.write)

c.perform()

# transform output into a dict:

response = json.loads(data.getvalue())

c.close()

**Sample output**

The request key contains various useful information about your request:

id

the request id

key

your private key

email

the email address you supplied when creating the request

target

the target Odoo version you supplied when creating the request

aim

the purpose (test, production) of your database upgrade request you supplied when creating the request

filename

the filename you supplied when creating the request

timezone

the timezone you supplied when creating the request

state

the state of your request

issue\_stage

the stage of the issue we have create on Odoo main server

issue

the id of the issue we have create on Odoo main server

status\_url

the URL to access your database upgrade request html page

notes\_url

the URL to get the notes about your database upgrade

original\_sql\_url

the URL used to get your uploaded (not upgraded) database as an SQL stream

original\_dump\_url

the URL used to get your uploaded (not upgraded) database as an archive file

upgraded\_sql\_url

the URL used to get your upgraded database as an SQL stream

upgraded\_dump\_url

the URL used to get your upgraded database as an archive file

modules\_url

the URL used to get your custom modules

filesize

the size of your uploaded database file

database\_uuid

the Unique ID of your database

created\_at

the date when you created the request

estimated\_time

an estimation of the time it takes to upgrade your database

processed\_at

time when your database upgrade was started

elapsed

the time it takes to upgrade your database

filestore

your attachments were converted to the filestore

customer\_message

an important message related to your request

database\_version

the guessed Odoo version of your uploaded (not upgraded) database

postgresql

the guessed Postgresql version of your uploaded (not upgraded) database

compressions

the compression methods used by your uploaded database

* JSON

{

"failures": [],

"request": {

"id": 10534,

"key": "Aw7pItGVKFuZ\_FOR3U8VFQ==",

"email": "john.doe@example.com",

"target": "8.0",

"aim": "test",

"filename": "db\_name.dump",

"timezone": null,

"state": "draft",

"issue\_stage": "new",

"issue": 648398,

"status\_url": "https://upgrade.odoo.com/database/eu1/10534/Aw7pItGVKFuZ\_FOR3U8VFQ==/status",

"notes\_url": "https://upgrade.odoo.com/database/eu1/10534/Aw7pItGVKFuZ\_FOR3U8VFQ==/upgraded/notes",

"original\_sql\_url": "https://upgrade.odoo.com/database/eu1/10534/Aw7pItGVKFuZ\_FOR3U8VFQ==/original/sql",

"original\_dump\_url": "https://upgrade.odoo.com/database/eu1/10534/Aw7pItGVKFuZ\_FOR3U8VFQ==/original/archive",

"upgraded\_sql\_url": "https://upgrade.odoo.com/database/eu1/10534/Aw7pItGVKFuZ\_FOR3U8VFQ==/upgraded/sql",

"upgraded\_dump\_url": "https://upgrade.odoo.com/database/eu1/10534/Aw7pItGVKFuZ\_FOR3U8VFQ==/upgraded/archive",

"modules\_url": "https://upgrade.odoo.com/database/eu1/10534/Aw7pItGVKFuZ\_FOR3U8VFQ==/modules/archive",

"filesize": "912.99 Kb",

"database\_uuid": null,

"created\_at": "2015-09-09 07:13:49",

"estimated\_time": null,

"processed\_at": null,

"elapsed": "00:00",

"filestore": false,

"customer\_message": null,

"database\_version": null,

"postgresql": "9.4",

"compressions": [

"pgdmp\_custom",

"sql"

]

}

}

**Downloading your database dump**

Beside downloading your migrated database using the URL provided by the [status method](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-status-method), you can also use the SFTP protocol as described in the [request\_sftp\_access method](https://www.odoo.com/documentation/10.0/reference/upgrade_api.html#upgrade-api-request-sftp-access-method)

The diffence is that you'll only be able to download the migrated database. No uploading will be possible.

Your database upgrade request should be in the done state.

Once you have successfully connected using your SFTP client, you can download your database dump. Here is a sample session using the 'sftp' client:

$ sftp -P 2200 user\_10534@upgrade.odoo.com

Connected to upgrade.odoo.com.

sftp> get upgraded\_openchs.70.cdump /path/to/upgraded\_openchs.70.cdump

Downloading /upgraded\_openchs.70.cdump to /path/to/upgraded\_openchs.70.cdump