Odoo ETL Documentation

Release 12.0

Anne Gentle

Sep 05, 2020

Contents:

CHAPTER 1

Description

ETL is an Odoo module developed to move data between databases easily. It can be used for data migration from different Odoo versions Not necessarily consecutives (as Openupgrade), data synchronization between Odoo databases, joining of Odoo databases, and also cleaning data from Odoo databases.

ETL is an abbreviation for extraction, transformation and loading. The module usually runs in a separate database but can also be installed in the target database.

ETL was migrated from Odoo V8/9 to V12 so it can be installed in odoo V12 and can move data between V8 V9 V10 V11 and V12.

There are advantages of using the ETL module such as the following:

- Can be used by functional consultants.
- Simple development, native Odoo methods.
- is an Odoo module.
- Works in most Odoo version 12.
- Multiple uses as mentioned earlier.

After installing ETL module, navigate to ETL's manager model and create a new manager with the following details:

- **Name** contains the name of your manager without any specific restriction for the name.
- **Target ID Type** is a selection field where you can choose *Source ID* or *Builded ID*. When set to *Source ID*, the record XML ID that will be used in the migration process will be according to the default source exported external ID. On the other hand, when set to *Builded ID*, the record XML ID that will be used in the migration process will be customized according to the prefix set later at an additional field.

  Source ID is recommended when performing a migration process.
- **Source Hostname** the source database host URL that is used to access the Odoo database from remote OS. i.e.: http://192.168.1.101. NOTE: For some unknown reason for me, the connection must be done with the ip address. Trying with the URL does not work. Maybe a nginx problem...
- **Source Port** the source database port that is used to access the Odoo database. i.e.: 80.
- **Source Database** The source database name.

- **Source Login** the username used to login to the source database from the login page. Make sure this user have full access to all the models.
- **Source Password** the password, what else?.
- **Source Language** the source database default language. It's recommended to keep the language as default (en_US).
- **Odoo Source Version** The source odoo major version. Default is 8.

  NOTE: This module was NOT Tested with source versions earlier than 8
- **Target Hostname** the target database host URL that is used to access the Odoo database from remote OS. i.e.: http://192.168.1.101.
- **Target Port** the target database port that is used to access the Odoo database. i.e.: 8069.
- **Target Database** the target database name.
- **Target Login** the username used to login to the target database from the login page. Make sure this user have full access to all the models.
- **Target Password** the password to access target system.
- **Target Language** the target database default language. It's recommended to keep the language as default (en_US).
- **Odoo Target Version the target odoo major version, default is the odoo** version where this module is installed

  NOTE: You can install this module in a odoo instance that is neither the source nor the target. In this case please be sure to change the Target Version for the target's odoo major version.
- **Models to delete Workflows**

CHAPTER 2

Notes Tab

If you started from a fresh target system you can define the modules to install in the **Notes** tab filling a python list of modules. You should use the technical names. With the button **Install Modules** you can install it and test target system connectivity. Besides you can write down some notes about the migration in the notepad field.

CHAPTER 3

Checking Source and Target connectivity

After filling all the Manager form data you can test connection to the source and target system with the button **Read models**, Then all the models from both system will be read (be sure to set suitable timeouts in the odoo.conf) You can see the models read in the **External models** tab.

CHAPTER 4

Read Databases

To read the models and get the record counts from the source and the target database click **READ AND GET RECORDS** from the action bar. The following two buttons make both actions individually so to speak, pressing **READ MODELS** and then **GET RECORD NUMBERS** is the same to press **READ AND GET RECORDS**

The ETL module will now attempt to connect and read from the source and destination databases.

After the process is done, the **External Models** tab from your manager form view should contain the list of models that have been read from the source and target database (along with its fields when clicked) and record counts.

CHAPTER 5

Mapping source and target models

Matching the source and target models along with its fields can be done automatically by ETL; however, the result may not be perfectly correct if you are moving data from different odoo versions.

Some models and fields change between odoo versions, in this case you may have to manually adjust the migration, this is explained in the next section. To perform an automatic mapping of models and fields, just click **Match and Order** in the action bar.

After the process is done, the Actions tab from your manager form view should contain the list of actions (model mappings) that have been matched and ordered by ETL. The models that etl could not match are flagged **To Analyze** the ones that cout match are flagged **Enabled**

After the actions have been generated, matched or not, it is necessary to order them since there are dependencies, some models depend on others and things must be executed in order. Just press **ORDER ACTIONS**

CHAPTER 6

Testing Actions (The hard work begins)

At the first use of the ETL manager, it's necessary to test the actions one by one which also means the migration will happen model by model for the first time. An action represents a migration for a single model at a time. Actions can also be understood as model mapping. It's not necessary to configure all the actions/model mapping implied by the Match and Order action, but only the required actions/model mapping necessary for the intended migration.

To be able to configure the actions and test it, simply click it from the list of actions in the manager.

Following are the details about the fields in the action model:

- **Name** the name of the action which is usually filled automatically for the **Match and Order** previous action.
- **Source Domain** used to apply domain for the source database model when performing the migration to filter out or include certain records in the migration.
- **Blocked** used to block the actions from running instead of having to switch the status to disabled. This field is used when configuring and testing the action on the first run of migration. After done configuring the action, Blocked field will usually be checked then later unchecked when performing the real migration which will be explained in the next section.
- **Sequence** used to set te execution order or the actions. The order in which actions (representing models) will be performed is really important due to the dependencies between models. i.e. The sequence of customer tags model should be lower than the customer model since customer model's migration will require the existing records of tags when the field tag_ids is enabled (configuration field will be explained in the next section). When actions are in list view it can be reordeed with drag and drop.
- **Repeating Action** is a read-only field which will be automatically checked when one of the fields states in the action's Field Mapping list is set to on_repeating. When this field is checked, the Run Repeated Action button will appear in the action bar of the Actions model form.
- **From Record** is also used to filter out or include records in the migration process. The ID records that will be migrated will start from the value set at this field. To disable this feature, simply leave it along with the To Record field to its default value 0 (zero).
- **To Record** is also used to filter out or include records in the migration process. The records that will be migrated will end at the value set at this field. To disable this feature, simply leave it along with the From Record field to its default value 0 (zero).

- **Source Model** the source model to migrate. The **Match and Order** action will try to fill this field.
- **source_id_exp** field name of the ID field in the source model. Usually set at its default (id).
- **Source Records** read only field counting the number of records at the source database in relation to the selected source model. The number of non-active records will not be counted, but can still be included in migration by setting the domain ['|', ('active', '=', True), ('active', '=', False)]
- **Target Model** the target model name which will be mapped to receive the records from the source model when running the action.
- **Target ID Type** have the same function as the manager model's *Target ID Type*. The default value will follow the value set at the manager model's Target ID Type and can be changed in every action according to preference (not recommended).
- **Target Records** read only field counting the number of records at the target database in relation to the selected target model. The number of the non-active records will not be counted.
- **target_id_prefix** field will only appear when the Target ID Type field is set to Builded ID allowing the customization of the records XML id instead of using the default export external ID.

The *Action* fields are usually set correctly by the automatic **Match and Order** action. Beside configuring the fields, it's very important to set the action's state which can be changed to the following possible states:

- **Enabled** The action will be included in the migration process.
- **To Analyse** The action requires further analysis and testing. and will not be included in the migration process.
- **Disabled** The action will not be included in the migration process.
- **No Records** The action will not be included in the migration process due to 0 records found in the source model.

After correctly configuring and checking the *Action* fields, it's very important to also check and configure every line of field mapping in the field mapping list in every actions. The field mapping determines which field of the selected model to be included or excluded in the migration process. To configure the fields, simply click the field mapping from the field mapping list of the action form.

Following is the details about the default fields in the field mapping model:

- **Blocked** works in a similar way with the Blocked field of the action model which in this case is used to block fields that have been analysed so that the data of this field is not included when running the testing.
- **Source Type** is a selection field which is set to the default value field for standard field data migration. Other source types will be explained in the next section.
- **Source Field** is the selection of source field names in respect to the selected source model in the action model form.
- **Source Exp.** is a short description of the selected source field.
- **Target Field** is the selection of target field names in respect to the selected target model in the action model form.
- **Target Exp.** is a short description of the selected target field.
- **Notes** is a field prepared for ETL users to write a longer notes for each field mapping.

Field mapping also have a state similar to that of actions and it's also very important to set the field mapping's state which can be changed to the following possible state:

- **Enabled** should be set to a field mapping that will be included in the action testing process and or migration process.
- **To Analyse** should be set to a field mapping that require a further analysis and testing. When a field mapping is set to this state, it will not be included when running the action testing process and or migration process.

- **Disabled** should be set to a field mapping that will not be included in the action testing process and or migration process.
- **Other Class** should be set to a field mapping that involves a relational field in which the record will be migrated from the other model.
- **On Repeating** should be set to a field mapping that usually involves a relational field that requires its own records such as parent/child relation or records from other models having a higher action sequence. This field data will be migrated after the first migration iteration by clicking **Run Repeated Action** in terms of action testing or **Run Repeated Actions** while running migration from the manager later after completing the action testing.

After configuring all the field mapping and the configuration for a specific action, test the action by clicking the **Run Action** button at the action bar for that specific action.

To see the result of the test, check the value of **Target Records**. If it increases after the process in regards to the **Source Records**, then the migration can be considered to be successful. To further confirm and check for errors, click the Log tab of the Actions form. When the test is successful, the logs will show an array of new created database id for the migrated records; otherwise, error messages will be shown. Address the error by reconfiguring the setting and field mappings of that specific action then re-run the test.

CHAPTER 7

Cleaning the Target and ETL Database

After correctly configuring and testing all the actions/model mapping, disable the other actions that will not be necessary for the intended migration and unblock all the actions. Before proceeding, do not forget to backup your ETL database.

Since the target database have been used for the testing, it's recommended to drop the database and recreate it. Make sure the modules are also installed again. If the target database name is changed, don't forget to change the Target Database at the ETL manager.

CHAPTER 8

Perform Migration

To perform the migration, simply click **Run Actions** button at the action bar of the manager form view. This will run all the actions according to our configuration in order. When process is completed, try checking for errors at every action's log since errors may still happen due to little misconfiguration.

When errors are found, try to address the errors accordingly by reconfiguring the fields then re-run the migration.

When no errors are found, click **Run Repeated Actions** button at the action bar of the manager form view as well to migrate the field mappings where state is set to On Repeating.

Re-check for error at the action logs and try to address them if there is one or more. After addressing the error, re-run the **Run Repeated Actions** action.

When no errors are found, migration can be considered to be successful.

CHAPTER 9

Manually Mapping

Manual mapping for both models and fields are possible when the automatic **Match and Order** action is inaccurate.

To manually map a model, navigate to the actions list view and create a new action/model mapping. Select the manager in the **Manager** field of the action then enter the detail of the action fields accordingly as described in Step 10 of the migration process. If the Source Model and the Target Model selection is empty, make sure the Manager field is set to the correct manager that have perform the **Read and Get Records** action.

After creating the action, click **Add an item** at the **Field Mapping** tab of that specific action to create the field mapping. Enter the detail of the field mapping fields accordingly as described in Step 10 of the migration process.

CHAPTER 10

Error Handling for Selection Fields and Value Mapping

Selection fields may cause confusing errors during migration since the source field valid selection values may be different with the target field valid selection values.

For example, in OpenERP version 7.0, the **priority** field of the project.task model have the following selection range: “Very Low”, “Low”, “Medium”, “Important”, “Very Important”. In Odoo version 9.0, however, the selection range of the same field allows a different selection range such as following: “Normal”, “High”.

In this case, we need to utilize ETL’s Value Mapping Fields.

> Source Field –> Value Mapping –> Target Field

CHAPTER 11

Value Mapping Fields

To use value mapping fields, navigate to the value mapping fields list view and click create. Set a name to the value mapping field at the **Field Name** field then set the type value to **Selection**. Set the **manager_id** field value to the specific manager that will be used for the migration.

For every possible selection values (both at source and at destination), create a **Mapping Value** record by clicking **Add an item** at the **Mapping Values** list. **Key** should be the real selection value and **Help Name** can be a short description for that specific selection value or simply the same value with **Key**.

For example, the **Mapping Values** for the **project.task priority** field will be as following:

Key	Help Name
Very Important	Very Important
Important	Important
Medium	Medium
Low	Low
Very Low	Very Low
Normal	Normal
High	High

After setting the Mapping Values, do not directly do the Details list. Click save, then edit to continue entering the Details list. The value mapping will be done in the Details list according to the Source Value and Target Value.

For example, the Details for the project.task priority field will be as following:

Source Value	Target Value
Very Low	Normal
Low	Normal
Medium	Normal
Important	High
Very Important	High

Click **Save** to save the **Value Mapping Fields** record. When the Value Mapping Field for a specific selection field is

have been created, navigate to the action containing that specific field mapping, click the intended field mapping, then set the **Source Type** field to **Value Mapping** and set the **Value Mapping Field** to the specific value mapping field record that have been created. Save the changes that have been made.

The value mapping example for the **project.task priority** selection field is shown according to the following image:

image

CHAPTER 12

Python Expression

Some field mappings may be enhanced with python code to allow more dynamic values at the target field. To utilize expressions field mapping, navigate to the field mapping that will the require the expression then changing the **Source Type** into **expression**. After setting the type into expression, an additional field expression will appear. The python expression will be coded inside this field.

Source Field –> Expression –> Target Field

Following python code from **field_mapping.py** located inside the ETL addons directory shows possible objects that can be accessed from the expressions:

```
@api.multi
def run_expressions(self, rec_id, source_connection=False, target_connection=False):
    result = []
    for field_mapping in self:
        expression_result = False
        if not source_connection or not target_connection:
            (source_connection, target_connection) = field_mapping.action_id.manager_
↪id.open_connections()
        source_model_obj = source_connection.model(field_mapping.action_id.source_
↪model_id.model)
        target_model_obj = target_connection.model(field_mapping.action_id.target_
↪model_id.model)
        obj_pool = source_model_obj
        cxt = {
                'self': obj_pool, #to be replaced by target_obj
                'source_obj': source_model_obj,
                'source_connection': source_connection,
                'target_obj': target_model_obj,
                'target_connection': target_connection,
                'rec_id': rec_id,
                'pool': self.pool,
                'time': time,
                'cr': self._cr,
                # copy context to prevent side-effects of eval
                'context': dict(self._context),
```

(continues on next page)

(continued from previous page)

```
                'uid': self.env.user.id,
                'user': self.env.user,
        }
        if not field_mapping.expression:
            raise Warning(_('Warning. Type expression choosen buy not expression set
↪'))
        # nocopy allows to return 'action'
        eval(field_mapping.expression.strip(), cxt, mode="exec")

        if 'result' in cxt['context']:
            expression_result = cxt['context'].get('result')
        result.append(expression_result)

    return result
```

For further details, please open field_mapping.py at the addons folder of the ETL module.

CHAPTER 13

Error Handling for Relational Field Using Raw Integer as ID

Some models such as **mail.followers** has a field such as res_id that stores the ID of the resource/record it attached to in a raw integer type (int) instead of relational type (many2one / one2many / many2many). Hence, when it's migrated, there will be no technical error but the res_id remains the resource ID of the source database which may change in the destination database. This error can be solved by using python expressions in the field mapping. Following is the python expressions used to solve this issue related to the **mail.followers res_id**:

```
source_ip = str(source_connection._server).split("://")[1].split(":")[0]
destination_ip =
str(target_connection._server).split("://")[1].split(":")[0]
source_db = str(source_connection).split("#")[1].split("'")[0]
destination_db = str(target_connection).split("#")[1].split("'")[0]
cr.execute("""SELECT destination.res_id as destination_res_id FROM
dblink('dbname='%s' port=5432 host='%s' user=leonardo
password=123','select a.id, a.res_model, a.res_id, b.name from
mail_followers a left join ir_model_data b on a.res_model = b.model and
a.res_id = b.res_id') AS source(id integer, res_model varchar, res_id
integer, name varchar), dblink('dbname='%s' port=5432 host='%s'
user=postgres password=123','select res_id, name from ir_model_data') AS
destination(res_id integer, name varchar) WHERE source.res_model in
(SELECT * FROM dblink('dbname='%s' port=5432 host='%s' user=postgres
password=123','select model from ir_model') AS model(model varchar)) AND
source.name = destination.name AND source.id = %s""",(source_db,
source_ip, destination_db, destination_ip, destination_db, destination_ip,
rec_id,))

try:
    context['result']= [r for r in cr.fetchall()][0][0]
except:
    context['result']= False
```

Do note that the above python code uses the **dblink** extension function from Postgres which require details such as database port, user, and password. In above case, source Postgres database have the following credential:

```
DB User     : leonardo
DB Password : 123
DB Port     : 5432
```

In above case, the destination Postgres database have the following credential:

```
DB User     : postgres
DB Password : 123
DB Port     : 5432
```

It is very crucial to execute the following SQL at the ETL's PostgreSQL database (not source or destination) before using the expressions containing the dblink Postgres function:

CREATE EXTENSION dblink;

CHAPTER 14

Error Handling for Create Date Field

ETL does not support the migration of the create and write date for all the Odoo models. After running the migration, create and write date will be set to the migration date. It is in fact that this create or write date field can be ignored in some modules, but for some other modules it may be crucial. In that case it's necessary to manipulate the create and or write date with python expressions to allow the accurate migration for create and or write date. Following is the python expression used to solve the create date issue related to the **crm.lead** model in which create date is crucial:

```
source_ip = str(source_connection._server).split("://")[1].split(":")[0]
destination_ip =
str(target_connection._server).split("://")[1].split(":")[0]
source_db = str(source_connection).split("#")[1].split("'")[0]
destination_db = str(target_connection).split("#")[1].split("'")[0]
cr.execute("""SELECT destination.id, source.create_date FROM
dblink('dbname='%s' port=5432 host='%s' user=leonardo
password=123','SELECT a.id, b.name, a.create_date FROM crm_lead a,
ir_model_data b WHERE a.id = b.res_id and b.model = ''crm.lead''') AS
source(id integer, name varchar, create_date timestamp),
dblink('dbname='%s' port=5432 host='%s' user=postgres
password=123','SELECT a.id, b.name, a.create_date FROM crm_lead a,
ir_model_data b WHERE a.id = b.res_id and b.model = ''crm.lead''') AS
destination(id integer, name varchar, create_date timestamp) WHERE
source.name = destination.name AND source.id = %s""",(source_db,
source_ip, destination_db, destination_ip, rec_id,))
matching_record = [r for r in cr.fetchall()][0]
dest_id = matching_record[0]
create_date = matching_record[1]
cr.execute("""SELECT dblink_exec('dbname='%s' port=5432 host='%s'
user=postgres password=123','UPDATE crm_lead SET create_date = TIMESTAMP
'%s' WHERE id = %s')""",(destination_db, destination_ip, create_date,
dest_id))
context['result']= str(create_date)
```

Do note that the above python code uses the dblink extension function from Postgres which require details such as database port, user, and password. In above case, source Postgres database have the following credential:

```
DB User     : leonardo
DB Password : 123
DB Port     : 5432
```

In above case, the destination Postgres database have the following credential:

```
DB User     : postgres
DB Password : 123
DB Port     : 5432
```

It is very crucial to execute the following SQL at the ETL's PostgreSQL database (not source or destination) before using the expressions containing the dblink Postgres function:

CREATE EXTENSION dblink;

CHAPTER 15

Error Handling for Many to Many Field Migration

The ETL module source code contains a bug related to the migration of many to many field type. This can be solved by modifying the action.py python script at line 471 located at the ETL addons folder.

Replace:

new_field_value = value

Into:

if new_field_value: new_field_value = new_field_value + ',' + value

else: new_field_value = value

CHAPTER 16

More about ETL's Migration Method

As mentioned earlier, one of the advantages of ETL is that it uses the native Odoo method. This can be found at the **action.py python** script at line 580 (unmodified action.py) located at the ETL addons folder.

ETL calls the load function of OpenERP to load the data into the target model. The load function can be found at the models.py python script starting at line 1022 (unmodified models.py at Odoo version 9) located at the OpenERP directory of Odoo.

CHAPTER 17

About the ERPpeek

Every connection made from the ETL database to the source and target database uses the methods from python library called ERPpeek in which ERPpeek itself uses xmlrpc to communicate with the databases. The source and target destination is called as a class object Client. Actions done at those databases are also done using methods from ERPpeek.

A <–> ERPpeek <–> ETL <–> ERPpeek

The ERPpeek python code can be viewed at the following link: https://github.com/tinyerp/erppeek/blob/master/erppeek.py.

Make sure the erppeek library is available in the server. sudo pip install erppeek

You can install odoo-etl on the target system, however you can install on an intermediate system with some benefits. That intermediate odoo would connect with the source and the target. That way the migration information will be saved in the intermediate system and can be used to move data from other systems. If the installation is done on the destination system once the migration is complete, the most natural thing would be to eliminate the odoo-etl module, which would erase all the know-how of the migration.

Is advisable to set workers=0 and limit_time_real = 1200 in the odoo.conf file to get rid of timeout problems and please, keep an eye open to the logs.

- INFINITY SOLUTIONS
- Ingadhoc by Juan Jose Scarafia <jjscarafia@adhoc.com>
- jeo Software by bJorge Obiols <jorge.obiols@gmail.com> (www.jeosoft.com.ar)

This sofware was originaly developed by INFINITY SOLUTIONS

It was migrated and improved by jjscarafia at ADHOC for Odoo V8/9 and actually deprecated the original project can be found at deprecated project

Migrated to v12 and mantained since 2019 by jobiols <jorge.obiols@gmail.com> at jeo Software. the project lives in project home and documentation can be found at project doc

CHAPTER 18

12.0.0.0.0 (2019-10-16)

- [MIG] migrate. (#3)

CHAPTER 19

9.0 (2015-??)

- [MIG] Support and migration

CHAPTER 20

8.0 (2014-??)

- [MIG] Support and migration
- Taking project at ADHOC by jjscarafia

CHAPTER 21

7.0 (????-??-??) Lost in time

- Starting proyect by Infinity Solutions

Hacer exportacion e importacion de scripts de migracion para poder ponerlos en git

- It is recommendend to delete all external identifiers on source database for model "res_partner" because when creating a user, odoo simulates partner creation and raise a unique constraint (except the admin user)
- Also could be recommendend to delete external identifiers related to product and product_temlate (except service products)
- Advisable to configure xmlrpc users to timezone zero to avoid errors
- Asegurarse de tener permisos manger para este modulo.
- Es aconsejable quitar las restricciones de timeout poniendo workers=0

CHAPTER 22

1- Crear un registro Manager

En este formulario se ponen los datos de Fuente y Destino de las instancias de odoo para las que vamos a trabajar.

CHAPTER 23

1- Leer los modelos

Con el boton **READ MODELS** se leen los modelos de las instancias Fuente y Destino y luego de la carga se pueden ver en la pestaña **External Models**

El boton **GET RECORD NUMBER** lee la cantidad de registros en cada modelo

Luego con el boton **MATCH MODELS** intenta machear los modelos y los campos de los modelos, creando **Acciones**

Las acciones aparecen en gris, verde o azul segun estan *deshabilitadas*, *habilitadas*, o *para analizar*

CHAPTER 24

Indices and tables

- genindex
- modindex
- search