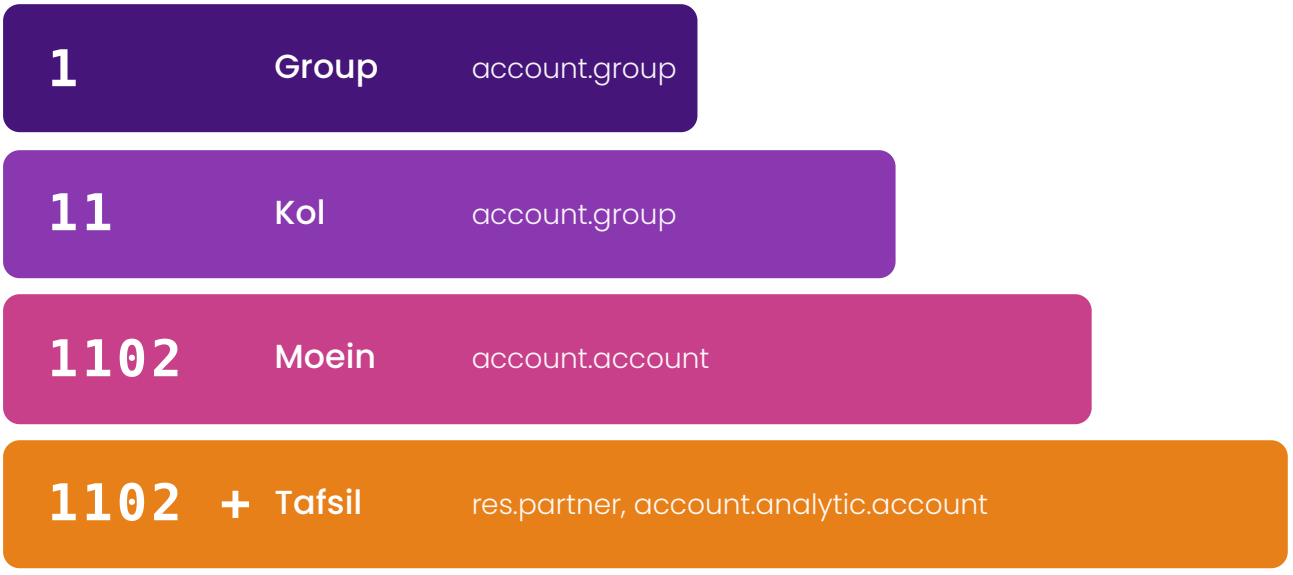
Training guide
Odoo 17, 18 and 19

Account coding in Odoo

Group, Kol, Moein and Tafsil: how the four levels relate, and how Odoo knows which child belongs to which parent.



A functional and technical walkthrough for teammates who do not come from accounting.

Powered by **Amir Akbari**

October 2026

The short version

If you only read one page, read this one. Everything after it is detail and examples.

Iranian accounting organises the chart of accounts in four levels: Group گروه, Kol کل, Moein معین and Tafsil تفصیل. Odoo does not have four tables for that. It covers the same four levels with three building blocks:

Level	Odoo building block	In one sentence
Group Kol	`account.group`	A folder. It only adds up what is inside it. Nobody ever posts money to a group.
Moein	`account.account`	The account itself. This is the only level a journal entry can be posted to.
Tafsil	`res.partner` and `account.analytic.account`	Not a level in the tree. It is a label stuck on a journal line that says *who* (partner) or *for what* (project, cost centre).

How does Odoo know which child belongs to which parent?

For groups and accounts, **nobody chooses the parent. Odoo reads the digits of the code.** Account `1` `1` `0` `2` sits under group `1` `1` because its code starts with 11, and group `1` `1` sits under group `1` because its prefix starts with 1. Change the code and the account moves.

For Tafsil it is the opposite: a person (or a default rule) picks the partner and the analytic account on each journal line, and each analytic account is assigned to a plan by hand.

What is in this guide

1 Accounting in five minutes
2 The four levels and where they live in Odoo
3 The data model on one page
4 `account.group` : Group and Kol
5 `account.account` : Moein
6 Tafsil: partners and analytic accounts
7 Who is whose child: every relation in one table
8 Worked example: one vendor bill, start to finish
9 Technical flow: what the code does, and when
10 Things that surprise people
11 Cheat sheet and self-check

Scope: Odoo 17, 18 and 19 (Community and Enterprise). Where versions behave differently, the text says so. The colours are consistent throughout: Group Kol Moein Tafsil.

1 Accounting in five minutes

You do not need to be an accountant to follow the rest. You need four ideas.

Every event is written twice

When something financial happens (we buy, we sell, we pay), Odoo records a **journal entry**. A journal entry has at least two lines. One side is called **debit**, the other **credit**, and the two sides must always be equal. That is all "double entry" means.

We pay 5,000,000 IRR of electricity from the bank. An expense appeared, and money left the bank:

Account	Debit	Credit
Electricity expense	5,000,000	
Bank Mellat		5,000,000

An account is a labelled bucket

Each line lands in exactly one **account**. An account answers the question "what kind of thing is this money?": cash, a bank account, money a customer owes us, electricity expense. Financial reports are simply the totals of these buckets.

Levels are zoom levels

Different people need different amounts of detail from the same lines:

Who is asking	What they want to see	Level
The CEO	"How much do we hold in current assets?"	Group
The finance manager	"How much of that is cash and bank?"	Kol
The accountant	"What is the balance of the Bank Mellat current account?"	Moein
Sales or a project manager	"How much does customer Alborz owe us?" or "What has the Tabriz project cost?"	Tafsil

Think of it as folders, files and tags

If you have ever organised files on a computer, you already understand the structure:

On your computer	In accounting	What it does
A folder	Group	Holds sub-folders. Shows a total.
A sub-folder	Kol	Holds files. Shows a total.
A file	Moein	The thing you actually write into.
A tag on what you wrote	Tafsil	Lets you filter the same content another way, across many files.

> **The tag idea is the one to remember.** A customer is not a file inside the "Receivables" folder. A customer is a tag, and the same customer tag can appear on lines in several different accounts (receivable, advance received, sales). That is why Tafsil is not part of the tree.

2 The four levels and where they live in Odoo

The Iranian four-level coding translated into Odoo, one level at a time.

Level	Question it answers	Example	Odoo model	Postable?
Group گروه	Which big family?	1 Current Assets	`account.group`	No
Kol کل General ledger	Which heading inside that family?	11 Cash and Banks	`account.group`	No
Moein معین Subsidiary ledger	Which exact account?	1102 Bank Mellat, Current	`account.account`	**Yes**
Tafsil تفصیل Detail	Who? For what?	Alborz Trading Co. Tabriz Warehouse project	`res.partner` `account.analytic.account`	On the line

Reading a code

The code carries the hierarchy inside it. Each extra digit zooms in one level:

1102 — Group **Current Assets**
everything whose code starts with 1

1102 — Kol **Cash and Banks**
everything whose code starts with 11

1102 — Moein **Bank Mellat, Current**
the account with code 1102

1301+ — Tafsil **Alborz Trading Co.**
no digits of its own, it rides on the journal line

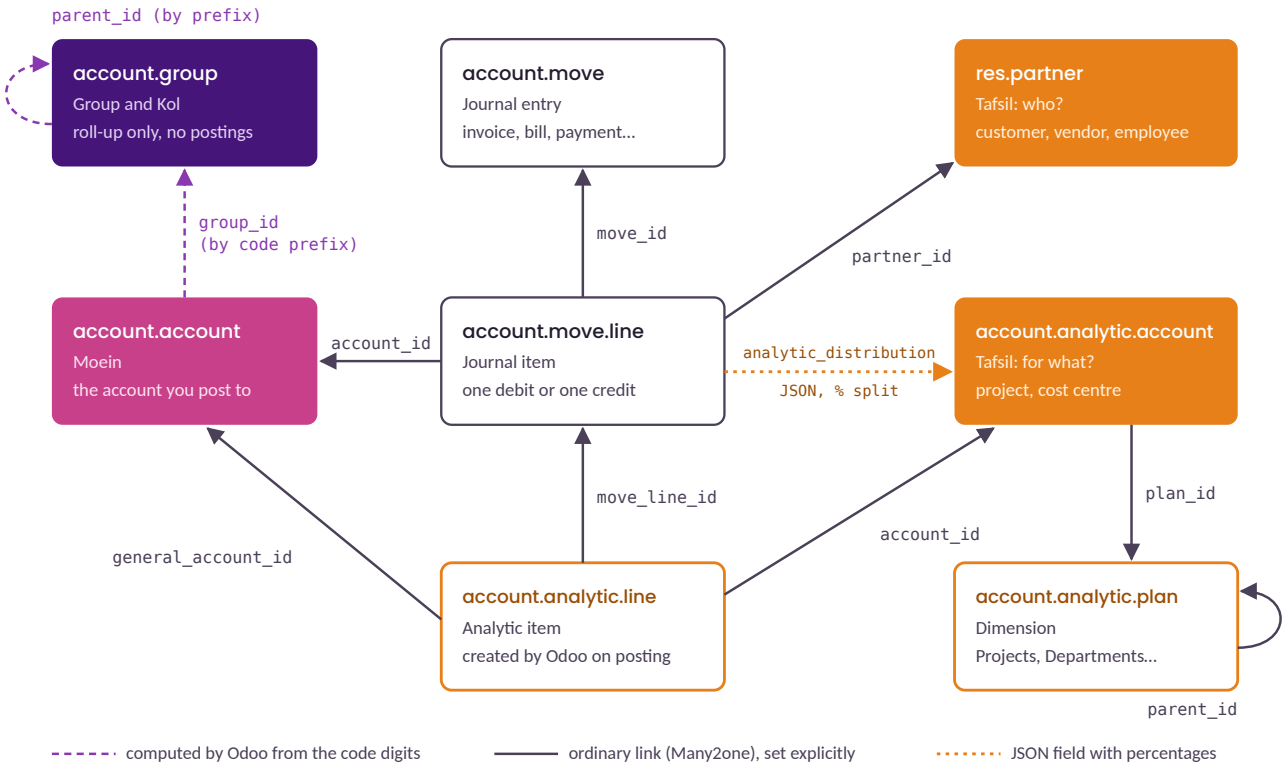
The example chart used in this guide

- 1 Current Assets — Group
  - 11 Cash and Banks — Kol
    - 1101 Cash on Hand — Moein
    - 1102 Bank Mellat, Current — Moein
  - 13 Trade Receivables — Kol
    - 1301 Accounts Receivable — Moein
      - Alborz Trading Co. — Tafsil
      - Sahand Steel — Tafsil
  - 16 Other Receivables — Kol
    - 1601 Input VAT — Moein
- 3 Current Liabilities — Group
  - 31 Trade Payables — Kol
    - 3101 Accounts Payable — Moein
      - Pars Electric Co. — Tafsil
- 8 Expenses — Group
  - 81 Operating Expenses — Kol
    - 8105 Repairs and Installation — Moein
      - Tabriz Warehouse — Tafsil
      - Tehran Office — Tafsil

> **The 1-2-4 digit pattern is our example, not an Odoo rule.** Any code length works, and groups can nest deeper: a group with prefix `1102` named "Banks" could hold accounts `110201` Bank Mellat and `110202` Bank Saman.

3 The data model on one page

Eight models, and how they point at each other. An arrow goes from the record that holds the link to the record it points at, so read it as "child points to parent".



How to read it

- The **left column** is the chart of accounts: groups on top, accounts below. The dashed purple links are never typed in by a user. Odoo works them out from the code.
- The **middle column** is what happens every day: a journal entry and its lines. Each line points at exactly one account.
- The **right column** is Tafsil. A line may point at one partner, and may be split across one or more analytic accounts. Analytic accounts are grouped into plans.
- The box at the bottom, `account.analytic.line`, is a by-product. When a journal entry is posted, Odoo writes one analytic line per analytic account so analytic reports are fast.

> **A naming detail that trips people up**
>
> The analytic model is `account.analytic.account` (singular "analytic", no "s"). Its database table is `account_analytic_account`. Model names use dots, table names use underscores.

4 `account.group` Group and Kol

The roll-up levels. A group is nothing more than a name and a range of code prefixes.

What it means in the business

- A group gives a sub-total. "Cash and Banks" shows the sum of every account whose code starts with 11.
- You cannot choose a group on a journal entry. Groups hold no money of their own.
- Both Group and Kol are stored in this same model. The only difference between them is the length of the prefix: a one-digit prefix is a Group, a two-digit prefix is a Kol.
- Where to find it: *Accounting ▸ Configuration ▸ Account Groups*. In some versions the menu shows only in developer mode.
- Where you see the result: the optional Group column of the Chart of Accounts list, and in Enterprise the Trial Balance and General Ledger with the option *Hierarchy and Subtotals* switched on.

What it is in the database

Field	Type	Meaning
`name`	Char	Label shown in reports, e.g. "Cash and Banks".
`code_prefix_start`	Char	First prefix covered by the group, e.g. `11`.
`code_prefix_end`	Char	Last prefix covered. Equal to the start for a single prefix; `13` to `15` would cover 13, 14 and 15.
`parent_id`	Many2one → `account.group`	The parent group. **Read-only: Odoo fills it in.**
`parent_path`	Char	The chain of ancestor ids, e.g. `1/2/`. Maintained by the ORM because the model has `_parent_store = True`; it makes "all descendants of X" searches fast.
`company_id`	Many2one → `res.company`	Groups belong to one company. Another company needs its own groups.

How Odoo finds the parent of a group

Every time a group is created or its prefix changes, Odoo re-evaluates the parents for that company using this rule:

> Among all groups of the same company whose prefix is **shorter** than mine and whose range **contains** my prefix, my parent is the one with the **longest** prefix. If none qualifies, I am a top-level group.

"Longest" matters when there are more than two levels: for a group with prefix 1102, both 1 and 11 contain it, and 11 wins because it is the closest ancestor.

id	name	start	end	parent_id, computed	parent_path
1	Current Assets	1	1	none, top level	1/
2	Cash and Banks	11	11	1, Current Assets	1/2/
3	Trade Receivables	13	13	1, Current Assets	1/3/
4	Other Receivables	16	16	1, Current Assets	1/4/
5	Current Liabilities	3	3	none, top level	5/
6	Trade Payables	31	31	5, Current Liabilities	5/6/
7	Expenses	8	8	none, top level	7/
8	Operating Expenses	81	81	7, Expenses	7/8/

The same rule as (simplified) SQL

```
-- for each child group, pick the closest containing group
SELECT DISTINCT ON (child.id) child.id, parent.id AS parent_id
  FROM account_group parent
  JOIN account_group child
    ON length(parent.code_prefix_start) < length(child.code_prefix_start)
   AND parent.code_prefix_start <= left(child.code_prefix_start, length(parent.code_prefix_start))
   AND parent.code_prefix_end   >= left(child.code_prefix_end,   length(parent.code_prefix_end))
   AND parent.company_id = child.company_id
 ORDER BY child.id, length(parent.code_prefix_start) DESC;   -- longest prefix first
```

Two rules Odoo enforces

- The start and end prefix of one group must have the same number of characters (`11` to `15` is fine, `1` to `15` is rejected).
- Two groups with prefixes of the same length may not overlap in the same company (you cannot have both `11–13` and `13–15`).

---

5 `account.account` Moein

The posting level. Every debit and every credit in the system ends up in one of these.

What it means in the business

- Where to find it: *Accounting* ▸ *Configuration* ▸ *Chart of Accounts*.
- The two decisions that matter when you create one are the **code** and the **type**.
- The **code** decides which group the account rolls up into, as described in the previous section.
- The **type** decides how the account behaves and which financial statement it appears in.

> **Type and group are two separate things**
> The **group** is presentation: it decides the sub-totals you see. The **type** is behaviour: it decides balance sheet or profit and loss, whether the balance carries over to next year, and whether the account works as a receivable or payable. An account with a wrong group gives an untidy report. An account with a wrong type gives a wrong report.

Family `internal_group`	Main values of `account_type`	Report
Asset	`asset_receivable` `asset_cash` `asset_current` `asset_prepayments` `asset_fixed` `asset_non_current`	Balance sheet
Liability	`liability_payable` `liability_current` `liability_non_current` `liability_credit_card`	Balance sheet
Equity	`equity` `equity_unaffected`	Balance sheet
Income	`income` `income_other`	Profit and loss
Expense	`expense` `expense_direct_cost` `expense_depreciation`	Profit and loss
Off-balance	`off_balance`	Neither

What it is in the database

Field	Type	Meaning
`code`	Char	The account number, e.g. `1102`. Unique per company.
`name`	Char	"Bank Mellat, Current Account".
`account_type`	Selection	Behaviour and report, see the table above.
`internal_group`	Selection, computed	The family, derived from the first part of `account_type`.
`reconcile`	Boolean	Lines can be matched against each other (invoice against payment). Always on for receivable and payable.
`group_id`	Many2one → `account.group`, computed	The closest group. **Read-only: derived from the code.**
`currency_id`	Many2one	Forces a currency on the account (useful for a USD bank account).
`tax_ids`	Many2many	Default taxes when the account is used on a line.
`company_ids`	Many2many	Odoo 18 and later: one account can be shared by several companies and have a different code in each. Odoo 17 has a single `company_id`.

> **There is no "parent account" field**
> People coming from other accounting software look for a parent selector on the account form. It does not exist. An account never has another account as its parent, and it does not store a link to a parent chosen by hand. The hierarchy is entirely group + code prefix.

How Odoo finds the group of an account

> Take every group of the company whose prefix range contains the first digits of my code. Out of those, my group is the one with the **longest** prefix. If nothing matches, `group_id` stays empty.

Code	Account	account_type	Groups that match	Result: group_id
1102	Bank Mellat, Current	`asset_cash`	1 and 11	11 Cash and Banks
1301	Accounts Receivable	`asset_receivable`	1 and 13	13 Trade Receivables
1250	Short-term Investments	`asset_current`	1 only, there is no group 12	1 Current Assets
8105	Repairs and Installation	`expense`	8 and 81	81 Operating Expenses
9001	Memo account	`off_balance`	none	empty

The row for 1250 is worth a second look: an account does not need a Kol to exist. It attaches to the nearest group that fits, in this case directly to the Group level, and shows up in the report beside the Kol groups.

Version difference

In Odoo 17 and earlier, `group_id` is a stored column on `account_account`, rewritten whenever codes or groups change. From Odoo 18, account codes are stored per company, so `group_id` is no longer a database column: Odoo computes it at read time for the current company. The matching rule is identical.

---

6 Tafsil: partners and analytic accounts

Odoo has no model called "Tafsil". This is the one place where the Iranian structure has to be translated rather than copied.

In Iranian software a Tafsil is a fourth code under the Moein, often a "floating" one that can be reused under several Moein accounts. Odoo reaches the same result with two labels on the journal line. Both are floating by nature: neither is tied to a single account.

Partner: the built-in Tafsil for people and companies

Every journal line has an optional `partner_id`. For receivables and payables, this replaces the whole Tafsil level:

- You keep **one** account 1301 "Accounts Receivable" for all customers, not one account per customer.
- Each line on 1301 carries the customer in `partner_id`.
- The *Partner Ledger* and *Aged Receivable* reports group those lines by partner. That is the Tafsil report.
- Technically: `account.move.line.partner_id` → `res.partner`. The partner form also stores which receivable and payable account to use for that partner (`property_account_receivable_id`, `property_account_payable_id`).

Analytic accounts: Tafsil for projects, cost centres and departments

Analytic accounting is a second, parallel way of slicing the same amounts. It never changes the balance sheet or the P&L; it only answers "for what?".

Model	Functional name	What it is
`account.analytic.plan`	Analytic plan	A dimension: "Projects", "Departments", "Cost Centres". Plans can have sub-plans through `parent_id`, which the user selects.
`account.analytic.account`	Analytic account	One value in a dimension: "Tabriz Warehouse". Key fields: `name`, `code`, `plan_id` (required), `partner_id`, `company_id`.
`analytic_distribution` field on `account.move.line`	Analytic distribution	A JSON value on the journal line: which analytic accounts, and what percentage each.
`account.analytic.line`	Analytic item	The result. Written by Odoo when the entry is posted, one per analytic account, with the amount already split.

What the JSON looks like

```
# 70% to analytic account id 7, 30% to id 8
{"7": 70.0, "8": 30.0}

# Odoo 17+: two plans at once. 100% to
# project 7 AND department 12 together
{"7,12": 100.0}
```

Keys are analytic account ids written as text. Values are percentages. They are not foreign keys in the database sense, which is why the diagram draws this link differently.

Where the hierarchy is

Analytic accounts have **no parent analytic account**. The tree is on the plans:



> **Which one should carry a given Tafsil?**
> If the detail is a person or company that we invoice, pay or owe, it is a **partner**. If it is something we want to measure cost or revenue for (a project, a branch, a vehicle, a department, a contract), it is an **analytic account**. A line can carry both at once.

Making Tafsil mandatory or automatic

Iranian systems often force a Tafsil on certain Moein accounts. Odoo does this with two configuration models:

Model	Use
`account.analytic.applicability`	Rules on a plan: optional, mandatory or unavailable, depending on the document type, an account prefix or a product category. "Plan Projects is mandatory on every account starting with 8" is one rule.
`account.analytic.distribution.model`	Defaults: "when the partner is X, or the product is Y, or the account starts with 81, pre-fill this distribution". The user can still change it on the line.

7 Who is whose child: every relation in one table

This is the direct answer to "how does the system know?". There are two kinds of link, and it helps to keep them apart.

Links that Odoo works out from the code

Child → parent	Field that holds the link	Decided by	How
Kol → Group group → group	`account.group.parent_id`	Odoo	The child's prefix falls inside the parent's prefix range, and the parent's prefix is shorter. Closest one wins.
Moein → Kol account → group	`account.account.group_id`	Odoo	The first digits of the account code fall inside the group's prefix range. Longest prefix wins.

Links that are set explicitly

Child → parent	Field that holds the link	Decided by	How
Journal item → entry	`account.move.line.move_id`	Odoo	Set when the line is created inside the entry.
Journal item → Moein	`account.move.line.account_id`	User or default	Chosen on the line, or proposed from the product, partner, journal or tax (see section 9).
Journal item → Tafsil partner	`account.move.line.partner_id`	User	Copied from the partner of the invoice or payment.
Journal item → Tafsil analytic	`account.move.line.analytic_distribution`	User or default	JSON of analytic account ids and percentages; may be pre-filled by a distribution model.
Analytic account → plan	`account.analytic.account.plan_id`	User	Selected on the analytic account form. Required.
Plan → parent plan	`account.analytic.plan.parent_id`	User	Selected on the plan form. Optional.
Analytic item → everything	`account.analytic.line`: `move_line_id`, `account_id`, `general_account_id`	Odoo	Written on posting from the distribution of the journal item.

What this means in practice

- Change the **code** of an account from 1102 to 1302 and it leaves "Cash and Banks" and joins "Trade Receivables", together with all its history. No other field needs touching.
- Change the **prefix** of a group and every account and sub-group underneath re-attaches on its own.
- Delete a Kol group and its accounts do not disappear; they fall back to the next matching group above (the Group level).
- Nothing of the kind happens for analytic: moving an analytic account to another plan is always a manual edit of `plan_id`.

8 Worked example: one vendor bill, start to finish

Pars Electric Co. invoices us 100,000,000 IRR plus 10% VAT for electrical installation. 70% of the work was for the Tabriz Warehouse project and 30% for the Tehran Office project.

What the accountant enters

A vendor bill with partner Pars Electric Co. and one line: 100,000,000 on account 8105, tax 10%, analytic distribution 70 / 30. Odoo builds the journal entry:

#	Moein account_id	Tafsil partner_id	Tafsil analytic_distribution	Debit	Credit
1	8105 Repairs and Installation	Pars Electric Co.	Tabriz Warehouse 70% Tehran Office 30%	100,000,000	
2	1601 Input VAT	Pars Electric Co.	none	10,000,000	
3	3101 Accounts Payable	Pars Electric Co.	none		110,000,000
	Total (debit = credit, so the entry balances)			**110,000,000**	**110,000,000**

Following line 1 up the tree

Journal item 1 (100,000,000 debit) → `account_id` → 8105 Repairs and Installation → `group_id` → 81 Operating Expenses → `parent_id` → 8 Expenses

The line itself stores only `account_id`. It knows nothing about group 81 or group 8. The two dashed steps are computed from the digits **8**, **81**, **8105**.

What gets written to the database when the bill is posted

Table	New rows	Content
`account_move`	1	The bill: partner, date, journal, state "posted".
`account_move_line`	3	The three lines above. Line 1 holds `analytic_distribution = {"7": 70.0, "8": 30.0}`.
`account_analytic_line`	2	−70,000,000 on Tabriz Warehouse and −30,000,000 on Tehran Office, both pointing back to line 1 and to account 8105. Costs are stored as negative amounts.
`account_account`, `account_group`	0	Nothing. The chart is master data and is not touched by postings.

One entry, three readings

Trial Balance with hierarchy

By Group, Kol, Moein

Level	Balance
8 Expenses	100,000,000
8 1 Operating	100,000,000
8 1 0 5	100,000,000
1 Current Assets	10,000,000
1 6 Other Rec.	10,000,000
3 Current Liab.	−110,000,000
3 1 Payables	−110,000,000

Partner Ledger

By Tafsil: who

Partner	We owe
Pars Electric Co.	110,000,000

Read from lines on payable account 3101, grouped by `partner_id`.

Analytic report

By Tafsil: for what

Analytic account	Cost
Tabriz Warehouse	70,000,000
Tehran Office	30,000,000

Read from `account.analytic.line`. VAT is not part of project cost, so line 2 has no distribution.

9 Technical flow: what the code does, and when

Three moments matter: saving a group, saving an account, and posting an entry. Method names are from the Odoo 17 source of the `account` and `analytic` modules; check them in the version you are working on.

A. A group is created or its prefix changes

1. **create() / write()**
   on `account.group`.
2. **_adapt_parent_account_group()**
   Runs the SQL from section 4 for the company and rewrites `parent_id` on every group that needs it.
3. **Parent store update**
   The ORM refreshes `parent_path`.
4. **Accounts follow**
   Odoo 17: `_adapt_accounts_for_account_groups()` rewrites the stored `group_id`. Odoo 18+: nothing to rewrite, it is computed when read.

B. An account is created or its code changes

1. **create() / write()**
   on `account.account` with a new `code`.
2. **_compute_account_group()**
   Depends on `code`. Looks for the group with the longest matching prefix in the company.
3. **Reports regroup**
   Existing journal items are untouched; they still point to the same account id. Only the roll-up changes.

C. A journal entry is posted

1. **The user confirms an invoice, bill or payment**
   `account.move.action_post()`, which calls `_post()`.
2. **Odoo checks the entry**
   Debits equal credits, the period is not locked, every line has an account. If an analytic plan is mandatory for a line and is missing, posting stops here (`_validate_analytic_distribution()`).
3. **The entry gets its number and state "posted"**
   From this moment the lines count in the General Ledger and Trial Balance.
4. **Analytic items are generated**
   `account.move.line._create_analytic_lines()` reads `analytic_distribution` and writes one `account.analytic.line` per entry in the JSON, amount = line amount × percentage, sign reversed.
5. **Reports read the result**
   Financial reports sum `account.move.line` grouped by account, then by group. Analytic reports sum `account.analytic.line`.

Where does the account on each line come from?

Users rarely type the Moein by hand. Odoo proposes it from master data:

Line	Account taken from	Field
Customer receivable	The partner	`property_account_receivable_id`
Vendor payable	The partner	`property_account_payable_id`
Sales line	The product, otherwise its category	`property_account_income_id` `property_account_income_categ_id`
Purchase or expense line	The product, otherwise its category	`property_account_expense_id` `property_account_expense_categ_id`
Tax line (VAT)	The tax's repartition lines	`account.tax.repartition.line.account_id`
Bank or cash line	The journal	`account.journal.default_account_id`

Looking at it yourself in the Odoo shell

```
# Moein -> Kol -> Group
acc = env['account.account'].search([('code', '=', '8105')], limit=1)
acc.group_id.name                 # 'Operating Expenses'
acc.group_id.parent_id.name       # 'Expenses'
acc.group_id.parent_path          # '7/8/'

# Group 1 and every group below it, at any depth (uses parent_path)
g1 = env['account.group'].search([('code_prefix_start', '=', '1')], limit=1)
env['account.group'].search([('id', 'child_of', g1.id)]).mapped('name')

# every account under Group 1: just use the prefix, like Odoo does
env['account.account'].search([('code', '=like', '1%')]).mapped('code')

# Tafsil on a posted journal item
line = env['account.move.line'].browse(LINE_ID)
line.partner_id.name              # 'Pars Electric Co.'
line.analytic_distribution        # {'7': 70.0, '8': 30.0}
[(l.account_id.name, l.amount) # [('Tabriz Warehouse', -70000000.0),
 for l in line.analytic_line_ids] #  ('Tehran Office', -30000000.0)]
```

10 Things that surprise people

The questions that come up in almost every implementation.

What you see	Why, and what to do
"I cannot find where to set the parent of an account."	There is no such field. Give the account a code that starts with the group's prefix.
A group shows no accounts in the report.	Its prefix does not match the start of any account code. Check for a typo, a leading space, or a start and end prefix with different lengths.
An account appears directly under the Group, beside the Kol groups.	No Kol group covers its second digit. Create the missing group or correct the code.
Odoo refuses to save a group.	Either start and end prefixes have different lengths, or the range overlaps another group of the same length.
Someone created one receivable account per customer.	Use one receivable account and let `partner_id` be the Tafsil. Hundreds of customer accounts make the chart unusable and break the standard reports.
"I added analytic accounts but the balance sheet did not change."	Correct. Analytic is a parallel view. It never changes the amounts in the general ledger.
After renumbering an account, last year's report looks different.	Grouping is computed from the current code, not stored per entry. The account took its whole history to the new group.
The groups exist in company A but not in company B.	Groups are per company. Create or import them for each company.
A report is right on sub-totals but the account sits in the wrong statement.	That is the `account_type`, not the group. Fix the type.
Analytic distribution is filled on a draft bill but no analytic items exist.	Analytic items are written on posting. Draft entries have none.

Design advice for a new chart

- Decide the digit pattern first (for example 1 digit Group, 2 digits Kol, 4 digits Moein) and keep every code to it. A consistent length is what makes prefix matching predictable.
- Create the groups before importing the accounts, so every account lands in the right place the moment it is created.
- Keep Moein accounts for the nature of the amount. Push "who" and "for what" to partner and analytic. A good test: if the list could grow every month (customers, projects, vehicles), it is Tafsil, not Moein.
- Use analytic applicability rules to force Tafsil where the client's policy requires it, instead of relying on users to remember.

11 Cheat sheet and self-check

One table to keep open while working, and five questions to test yourself.

Persian	English	Odoo model	Database table	Parent found by
گروه	Group	`account.group`	`account_group`	Top level
کل	Kol	`account.group`	`account_group`	Code prefix
معین	Moein	`account.account`	`account_account`	Code prefix
تفصیل	Tafsil who	`res.partner`	`res_partner`	Set on the line
تفصیل	Tafsil for what	`account.analytic.account`	`account_analytic_account`	Set on the line
سند حسابداری	Journal entry	`account.move`	`account_move`	
آرتیکل سند	Journal item	`account.move.line`	`account_move_line`	Its entry
	Analytic plan	`account.analytic.plan`	`account_analytic_plan`	Set by user
	Analytic item	`account.analytic.line`	`account_analytic_line`	Odoo, on posting

Check yourself

	Question	Answer
1	Groups 1, 11 and 13 exist. You create account 1405. Which group does it get?	Group 1. Nothing covers prefix 14, so it attaches to the closest group that fits.
2	Where do you set the parent of account 8105?	Nowhere. Its code starts with 81, so it belongs to group 81.
3	A new customer signs a contract. Do you create a new account?	No. You create a partner. Their invoices post to the shared receivable account with the partner on the line.
4	A bill is split 50/50 between two projects. How many journal items and analytic items does the expense line produce?	One journal item (with a two-entry distribution) and two analytic items.
5	You rename group 11 from "Cash and Banks" to "Liquid Funds". What moves?	Nothing moves. The name is only a label; the prefix decides membership.

> **The whole guide in two lines.** Group, Kol and Moein are connected by the digits of the code, and Odoo does the connecting. Tafsil is a label on the journal line, and a person or a default rule does the connecting.