

Marketing

Promotional codes, vouchers, personal promotions, subscriptions and wallet credit.

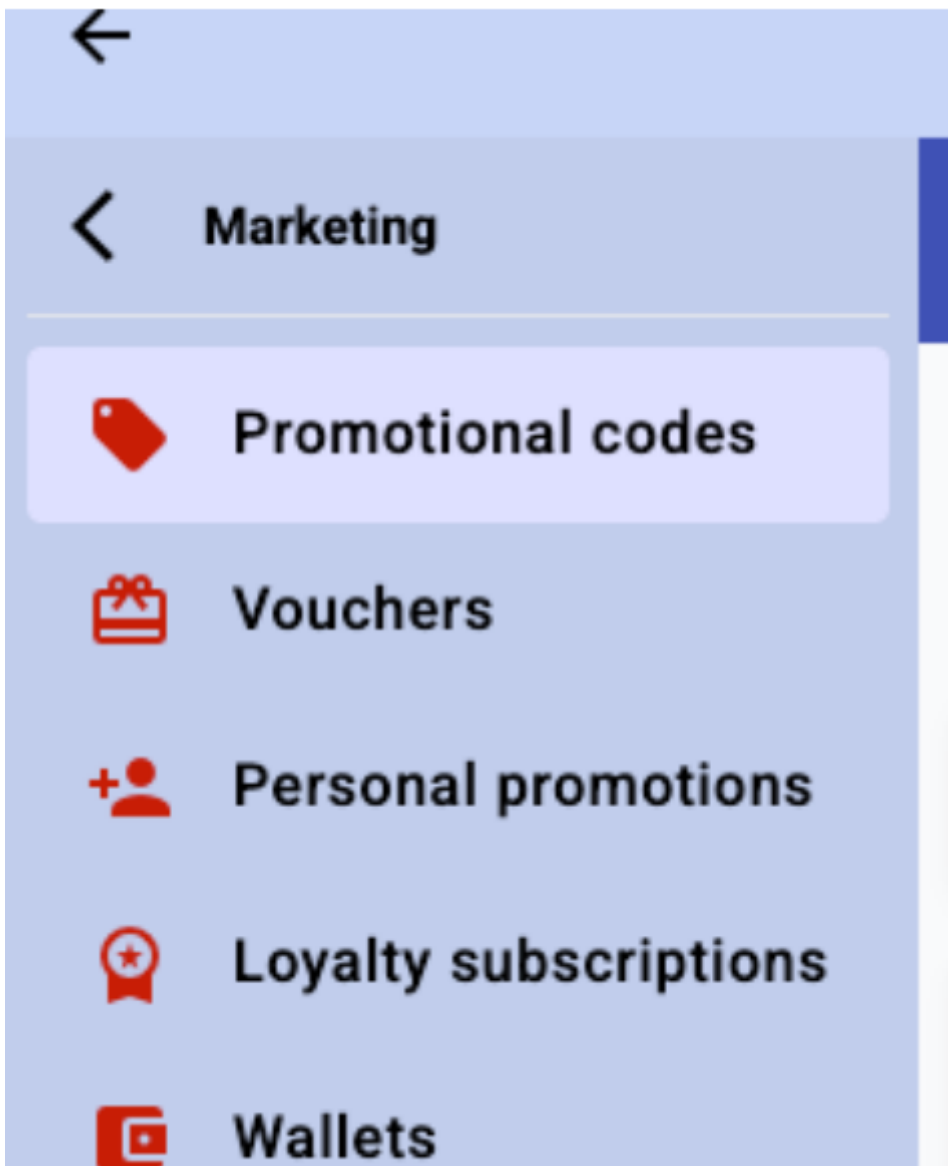
- [The Marketing screen](#)
- [Marketing: the five areas](#)
- [How bulk changes work](#)
- [Importing from a CSV](#)
- [Promotional codes](#)
- [Importing promotional codes](#)
- [Vouchers](#)
- [Personal promotions](#)
- [Loyalty subscriptions](#)
- [Wallets: finding a customer](#)
- [Wallets: managing records](#)

The Marketing screen

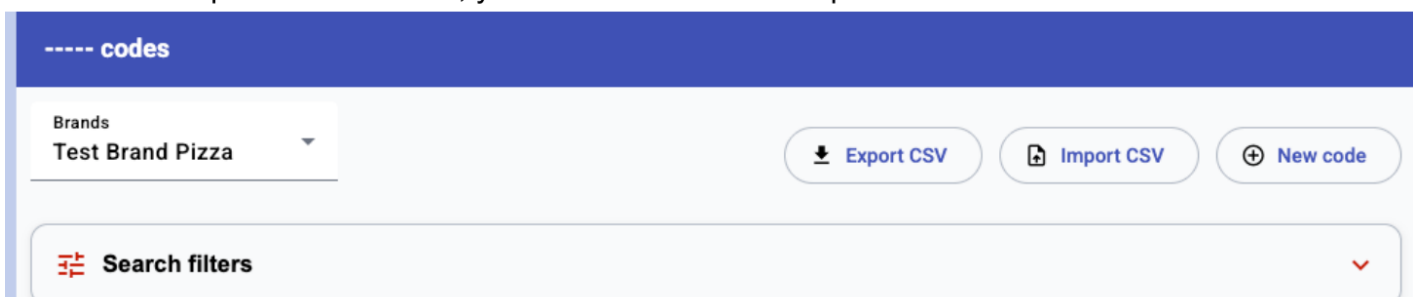
To access the code management section, go to the **MARKETING** menu in the console's central panel.



In the sidebar, you'll find all the sections available for viewing, editing, creating, or expiring codes: Promotional codes, Vouchers, Personal promotions, Loyalty subscriptions, and Wallets.



Each section loads a table containing the data for those codes. At the top of each section, you'll find the available options:



- **Brands:** If you have multiple brands, a dropdown will appear with the list of brands, and you must select the appropriate one for your query. If you only have one brand, this dropdown will not appear.
- **Export CSV:** Exports the data from the table below in CSV format.

- **Import CSV:** Loads a section where you can import a CSV file with the new codes you want to use. Each section has different specifications depending on the type of codes being imported.

“ Only CSV files are accepted, and they must strictly comply with the specifications.

CSV specifications and import

Selecting **Show CSV specifications** displays all the necessary information for that section on screen: how the first row of the CSV must be formatted, followed by a table with the required fields, a description of each field, and the allowed values.

CSV specifications

 The CSV must contain a header row followed by a single column, with one ----- code per row.

Field	Description	Allowed values
Code	Individual ----- code (one row per code, no duplicates).	5-50 characters. Only uppercase letters (A-Z) and numbers (0-9).

 Understood

The **SELECT CSV** button is only enabled once all required fields (if any) have been filled in.

Once a file is selected, if it passes validation (meets all requirements), a "**File successfully validated**" message appears next to the file, and the **Upload CSV** button becomes available.

**SELECT CSV**
Click to select a file

 test-codes.csv

 File successfully validated.

Total codes: 606

Upload CSV

Once the import begins, a message shows the status of the process, and a confirmation message appears when it finishes.

If the selected file fails validation, a **"Validation Error"** message appears next to the file, followed by a list of all errors found. At that point you can only select another file with those errors corrected — the upload button stays disabled until validation succeeds.


SELECT CSV
Click to select a file

 **Test_ERRORS.csv**⚠️ **Validation Error**

 **Could not read the CSV file. Error: Some codes are not valid. They must be between 5 and 50 characters and only A-Z or 0-9, please review the file and try again**

Other available actions

- **New code/voucher/...**: Used to create a single record in that section. Each section has its own specifications; if not all required fields are completed, the save button will not be enabled.
- **Search filters**: Each section has different search filters; some require filling in several fields to filter by that criterion. Filters can be reset at any time.

 **Search**

 **Clear filters**

Marketing: the five areas

“ **Audience:** marketing and customer-care staff who issue promotions, vouchers, subscriptions and wallet credit. **You need:** a permission per area — they are separate, and holding one grants nothing in the others.

Marketing covers five independent areas. They look similar and behave similarly, but each has its own data, its own permission, and its own rules.

Area	What it holds	Permission
Promotional codes	Campaigns and the coupons generated from them	<i>Promotional codes — view / edit</i>
Vouchers	Single-use vouchers of a fixed amount	<i>Voucher manager — view / edit</i>
Personal promotions	Promotions attached to specific customers	<i>Customer promo manager — view / edit</i>
Loyalty subscriptions	Customer subscriptions	<i>Subscriptions manager — view / edit</i>
Wallets	Customer wallet credit and its movements	<i>Wallet manager — view / edit</i>

Each area also has a **CSV import** permission of its own, separate again from view and edit. Someone who can edit wallet records cannot necessarily import wallet credit in bulk.

Finding the screens

All five live under the Marketing menu, each with its own section in the sidebar and the same row of options at the top — brand selector, export, import. [The Marketing screen](#) walks through that layout with screenshots, including the CSV specification panel each section provides.

Two things shared by all five

Before using any area, read these two pages. They explain mechanics that repeat everywhere and cause most of the confusion:

- [How bulk changes work](#) — the check → apply → track pattern, and the rule that protects already-redeemed records.
- [Importing from a CSV](#) — what the importers accept and how failures behave.

The rule that makes this safe

Across every area, bulk operations act only on records that have **not been redeemed**. A voucher a customer has already used, a wallet credit already spent, a coupon already applied to an order — these are never touched by a bulk update or a bulk delete.

This is what makes bulk changes recoverable in practice: the worst case is that you change records nobody has used yet.

“ The one place this bites is when the count comes back smaller than you expected. That usually isn't a filter problem — it's redemptions being excluded, exactly as intended.

Picking the right area

You want to...	Area
Run a campaign with codes customers enter	Promotional codes
Give one customer a fixed-value voucher	Vouchers
Attach an offer to specific customers	Personal promotions
Manage a customer's subscription	Loyalty subscriptions
Add or adjust customer credit	Wallets

These overlap less than they appear. A voucher is a fixed amount; a promotional code is a campaign rule; wallet credit is a balance. Choosing by *what the customer receives* gets it right nearly every time.

The pages in this chapter

Topic	Page
The shared bulk mechanics	How bulk changes work
The shared import mechanics	Importing from a CSV
Campaigns, coupons, offers	Promotional codes
Importing promotional codes	Importing promotional codes
Fixed-value vouchers	Vouchers
Customer-specific offers	Personal promotions
Subscriptions	Loyalty subscriptions
Finding a customer's wallet	Wallets: finding a customer
Adding and adjusting credit	Wallets: managing records

What you can and can't see

Everything here is scoped to the brand you are working in, and filtered further by your own scope — see [How scope works in practice](#).

How bulk changes work

The five Marketing areas are introduced in [Marketing: the five areas](#).

Wallets, vouchers, personal promotions and promotional codes all support changing many records at once. They follow the same three-step shape, and learning it once covers all four.

The three steps

Step	What it does	Permission
1. Check	Counts what <i>would</i> be affected. Changes nothing	The area's view permission
2. Apply	Starts the change as a background job	The area's edit permission
3. Track	Follows the job until it finishes	The area's edit permission

The check step is deliberately harmless and deliberately separate. It exists so you can see the blast radius before committing to it.

“ **Always run the check.** It takes seconds and it is the only preview you get. A count that looks wrong — far larger or far smaller than expected — means your criteria are wrong, and that is much cheaper to discover now.

Step 2 runs in the background

Apply does not finish while you wait. It returns a job that runs on its own, and you follow its progress.

This matters in three ways:

- **Closing the screen doesn't stop it.** The job carries on.
- **Progress is polled**, so the figure you see may lag slightly behind reality.
- **A job that is still running is not a job that failed.** Give a large change time before assuming something is wrong.

Already-redeemed records are never touched

Every bulk operation acts only on records that have **not been redeemed**.

Area	What is protected
Wallets	Credit already spent
Vouchers	Vouchers already used
Personal promotions	Promotions already applied to an order
Promotional codes	Coupons already redeemed

You cannot retroactively change or remove value a customer has already received. This is why bulk operations are recoverable in practice, and it is also the usual explanation for a count being smaller than expected.

Delete and update are separate operations

Bulk **update** changes fields — typically dates or amounts — on matching records. Bulk **delete** removes them.

Both are protected by the same non-redeemed rule, and both have their own check step. They are not variants of one action: pick the one you mean.

Promotional codes work differently — read this

Promotional codes use the same three words for a materially different operation.

In wallets, vouchers, personal promotions	In promotional codes
Check counts matching records	Check asks whether the campaign has <i>any</i> redeemed coupons

In wallets, vouchers, personal promotions	In promotional codes
Apply updates matching records	Apply deletes the whole campaign and its coupons — and only if none has been redeemed

So in promotional codes, "apply bulk changes" is a **delete**, gated on the campaign being entirely unused. If even one coupon has been redeemed, it will not proceed.

To retire a campaign that *has* been used, expire it instead — see [Promotional codes](#).

“ This is the single most dangerous naming overlap in the chapter. In four areas "apply" means update; in promotional codes it means delete a campaign. Check which screen you are on before clicking it.

A safe sequence

- 1. Set your criteria narrowly.
- 2. **Check**, and read the count.
- 3. If the count surprises you, stop and work out why.
- 4. Apply.
- 5. Track to completion.
- 6. Spot-check a couple of affected records.

If a job seems stuck

Give it time proportional to the count — tens of thousands of records take a while. If progress genuinely stops advancing, don't re-run the apply: a second job on the same criteria can compound the first. Report it with the area, the criteria and roughly when you started.

Importing from a CSV

Four of the areas described in [Marketing: the five areas](#) accept a CSV, so you can issue something to many customers at once.

Import	What it creates	Permission
Personal promotions	A promotion per customer, from a phone list	<i>Personal promotions — import</i>
Loyalty subscriptions	A subscription per customer, from a phone list	<i>Loyalty subscriptions — import</i>
Wallet credit	Credit per customer, from a phone list	<i>Wallet codes — import</i>
Vouchers	Voucher codes	<i>Voucher codes — import</i>

Promotional codes also import from CSV, but with progress tracking the others don't have — it gets its own page, [Importing promotional codes](#).

“ Each import permission is **separate** from the area's view and edit permissions. Being able to create wallet records one at a time does not imply being able to load thousands from a file.

Phone lists

Three of the four identify customers by **phone number**. The file is a list of phones plus whatever the import needs — an amount, a set of dates.

Phone numbers are where these imports go wrong. Before importing:

- **Check the format matches** how phones are stored for this brand — country prefix or not, spaces or not. A mismatch means rows silently match no customer.
- **Watch for spreadsheet damage.** Excel and Google Sheets both like to drop leading zeros and turn long numbers into scientific notation. A column formatted as *text* survives; a column formatted as *number* may not.
- **De-duplicate.** The same phone twice usually means that customer gets the benefit twice.

Before any import

1. **Start with a handful of rows.** Import five, confirm they landed as intended, then import the rest. This catches format problems for the price of five records instead of five thousand.
2. **Know what happens to an existing customer.** Whether a second import adds to what they have or replaces it decides whether re-running a file is safe.
3. **Keep the file.** It is the only record of exactly what you sent, and the only way to reverse it if you need to.

Imports are not transactional

An import that fails partway through does not roll back what it already did. Some customers will have received the benefit and others won't.

That is why the small test import matters, and why re-running a whole file after a partial failure is risky: the rows that already succeeded may be applied a second time.

If an import fails partway, work out **where** it stopped before deciding what to do — don't simply re-submit.

After importing

Spot-check two or three customers from the file and confirm they received what you intended, with the right amount and the right dates. The area's own browse screen is the place to do it: [Personal promotions](#), [Loyalty subscriptions](#), [Wallets: finding a customer](#) or [Vouchers](#).

If you need to undo one

There is no "undo import". What you have instead is a bulk delete, which removes matching **non-redeemed** records — see [How bulk changes work](#).

Anything customers already redeemed stays. This is the honest limit of reversing an import: you can take back what hasn't been used, not what has.

Promotional codes

“ **You need:** *Promotional codes — view* to browse, *Promotional codes — edit* to create or change anything.

Promotional codes are organised as **campaigns** that generate **coupons**. Getting that relationship right makes the rest of the screen obvious.

Term	What it is
Promotional code	The rule: what the discount is, when it is valid
Campaign	A promotional code plus the set of coupons generated from it
Coupon	One individual code a customer actually enters

A campaign is created with its first coupon. You then add further coupons to it as needed.

Browsing

The main list supports filters and pagination. From a campaign you can open its **generated coupons** — the individual codes — and see or edit them one at a time.

Active offers is a separate view: the offers currently live for a given store, with the detail of any one of them. Use it to answer "what can this store actually offer right now", which is not the same question as "what campaigns exist".

Creating

To create	Use
A standalone promotional code	Create promotional code
A campaign, with its first coupon	Create campaign
Another coupon on an existing campaign	Add coupon

Create the campaign first, then add coupons to it. Creating separate codes that were meant to be one campaign leaves you unable to manage them together afterwards.

Editing

Two different edits, at two levels:

- **The promotional code** — its dates, renewal day, and the times it is valid.
- **A single generated coupon** — one specific code within a campaign.

Editing the code changes the rule for the campaign. Editing a coupon changes one customer-facing code. If a customer reports a problem with one code, the coupon is usually the right level.

Expiring versus deleting

This is the decision that matters most, and the two are not interchangeable.

	Expire	Delete
What it does	Sets the end date to today — the code stops working from now	Removes the campaign and its coupons entirely
Redeemed coupons	Fine — history is preserved	Blocked. Will not proceed if any coupon has been redeemed
Reversible	The record still exists	No
Use when	A live campaign should stop	A campaign was created in error and never used

Expiring is almost always the right answer. It stops the campaign while keeping the record of what happened, which is what you need when a customer asks why their code stopped working.

There is also a bulk expire, which marks **every coupon of a campaign as used** in one go. Use it to close a campaign out completely rather than letting unredeemed coupons linger.

“Deleting a campaign is the operation described under "apply bulk changes" on this screen — a naming overlap that means something different everywhere else in Marketing. Read [How bulk changes work](#) before using it.

If a customer's code doesn't work

Check in this order:

1. **Is the campaign still within its dates?** Including the valid times, not just the days.
2. **Has that specific coupon already been redeemed?** Open the generated coupons and look.
3. **Is the offer live for that store?** Check active offers for that store — a campaign can exist without being offered everywhere.
4. **Was it bulk-expired?** A campaign closed out has all its coupons marked used.

Related pages

- [Importing promotional codes](#) — loading codes from a file
- [How bulk changes work](#) — before any bulk operation here
- [Personal promotions](#) — for offers attached to specific customers instead

Importing promotional codes

“ **You need:** *Promotional codes* — *edit*.

Promotional codes are the one import with full job tracking. Unlike the other CSV imports in [Importing from a CSV](#), you can see a running import's progress, and you get told if one is already going.

One active import at a time, per person

The import tracks an **active job for you**. Before starting, the screen tells you whether you already have one running.

This is a guard rather than a limitation. It stops the common accident of starting an import, seeing no immediate feedback, and starting it again — which would apply the same file twice.

What you see	What it means
No active import	Clear to start
An active import	Yours is still running. Watch it rather than starting another
An active import you don't recognise	Something you started earlier and left. Check its progress before clearing it

Running one

1. Check no import is already active for you.
2. Upload the CSV.
3. Follow the progress as it runs.
4. Confirm the result before telling anyone the codes are live.

Progress

The progress figure is polled, so it lags slightly — a number that hasn't moved for a few seconds is normal. Large files take proportionally longer.

Closing the screen does not stop the import. It runs to completion on its own, and you can come back to its progress.

Clearing a stuck active job

You can clear your active import. What that means is worth being precise about:

“ Clearing removes the **record that you have an import in progress**. It does not undo rows already imported, and it isn't a cancel button. Use it when a job is genuinely finished or abandoned and is blocking you from starting a new one — not to stop a running import you've changed your mind about.

If you clear a job and start the same file again, the rows that already imported may be applied twice. Check what landed first.

Before importing

The general advice in [Importing from a CSV](#) applies here too, particularly:

- **Test with a few rows first.** Confirm the format is accepted before committing a large file.
- **Keep the file.** It is your record of exactly what was issued.
- **Imports are not transactional.** A failure partway leaves earlier rows in place.

After importing

Open the promotional codes list and confirm the codes exist with the dates and validity you expected — see [Promotional codes](#). If they need to be part of a campaign, confirm that too, because a campaign relationship is hard to add afterwards.

If it goes wrong

Removing imported codes uses the campaign delete described in [Promotional codes](#), which only proceeds when **nothing has been redeemed**. Once customers start using codes, they stay — expire the campaign instead.

Vouchers

“ **You need:** *Voucher manager — view* to browse, *Voucher manager — edit* to create or change. Importing needs *Voucher codes — import*, which is separate again.

A voucher is a fixed amount a customer can redeem once. Simpler than a promotional code — there is no campaign, no rule, just a value and a validity period.

Browsing

The list shows **available vouchers** — those not yet redeemed — for the brand.

“ Redeemed vouchers are not in this list. That is deliberate: the screen is for vouchers still in play. If you are looking for one a customer says they used, its absence here is the answer.

Creating one

Create a voucher with its amount and validity. This is the right path for a single customer — a service recovery, a goodwill gesture, a one-off.

For many vouchers at once, import them instead — see [Importing from a CSV](#).

Editing

You can update an existing voucher: its amount, its dates. Only vouchers that haven't been redeemed can be changed, which is the same rule that protects bulk operations.

Expiring

Expiring ends one or more vouchers immediately. Use it when a voucher was issued in error, or when a promotion is being pulled.

Expiring is the safe option: the record survives, so there is still an answer when the customer asks what happened.

Bulk changes

Vouchers support the standard bulk pattern — read [How bulk changes work](#) first.

Voucher bulk operations match on **starting date, ending date, amount and validity days** together. All four form the criteria; they are not filters you can partly fill in and hope for the best.

Operation	Effect
Check	Counts non-redeemed vouchers matching all four criteria
Bulk update	Updates every matching non-redeemed voucher
Bulk delete	Removes every matching non-redeemed voucher

Because the criteria are a combination, it is easy to match more than you meant. Run the check and read the count — it is the only preview.

“ Only non-redeemed vouchers are ever affected. A customer who already used theirs keeps the benefit, whatever you do here.

Choosing between a voucher and wallet credit

They feel similar and behave differently:

	Voucher	Wallet credit
Shape	A fixed amount, used once	A balance that can be spent down

	Voucher	Wallet credit
Good for	A specific gesture with a clear value	Ongoing credit, refunds, loyalty balance
Expiry	Per voucher	Per credit record

If the customer should have "€10 off one order", that is a voucher. If they should have "€10 to spend", that is wallet credit — see [Wallets: managing records](#).

Related pages

- [How bulk changes work](#) — before any bulk operation
- [Importing from a CSV](#) — issuing vouchers in bulk
- [Wallets: managing records](#) — the alternative shape of credit

Personal promotions

“ **You need:** *Customer promo manager* — *view* to browse, ... — *edit* to change.
Importing needs *Personal promotions* — *import*, separate again.

A personal promotion attaches an existing offer to a **specific customer**. Where a promotional code is something any customer can enter, this is granted to one person.

What you can attach

The **available promos** view lists what can be attached for a given store. It is deliberately not the full catalogue — it shows offers that are:

- **hidden** — not publicly advertised, meant to be granted individually;
- **manually added** — designed for exactly this purpose;
- **not expired**.

“ If an offer you expect isn't in the list, it is usually one of those three: it is a public offer rather than a grantable one, or it has expired. That is the list doing its job, not a missing record.

Browsing existing records

Search customer promo records to see what a customer already has, or to review what was granted in a campaign of individual grants.

Granting one

1. Find the offer under available promos for the relevant store.
2. Create the record against the customer.
3. Set the dates it is valid between.

For many customers at once, import a phone list instead — see [Importing from a CSV](#).

Changing and removing

Action	Applies to	Note
Update dates	Non-redeemed records only	Extend or shorten validity
Delete	Non-redeemed records only	Removes the grant entirely

Both are limited to records the customer hasn't used. Once redeemed, a personal promotion is part of that customer's history and stays.

To withdraw an offer a customer has *not* used, shortening the dates is gentler than deleting: the record survives, so there is an answer if they ask.

Bulk changes

Personal promotions follow the standard pattern — read [How bulk changes work](#) first.

Here the criteria are a **promo or offer code**: you are acting on every record granting that particular offer.

Operation	Effect
Check	Counts records matching the promo/offer code
Bulk update	Changes the start and end dates on every matching non-redeemed record
Bulk delete	Removes every matching non-redeemed record

This is how you correct a grant campaign that went out with the wrong dates, without touching customers who already used theirs.

Choosing between this and a promotional code

	Personal promotion	Promotional code
--	--------------------	------------------

Who can use it	One named customer	Anyone with the code
How they get it	Granted to them	They enter a code
Good for	Win-back, VIP treatment, service recovery	Public campaigns

If you find yourself granting the same offer to hundreds of customers individually, a campaign with coupons is probably the better tool — see [Promotional codes](#).

Related pages

- [How bulk changes work](#) — before any bulk operation
- [Importing from a CSV](#) — granting from a phone list
- [Promotional codes](#) — the public-campaign alternative

Loyalty subscriptions

“ **You need:** *Subscriptions manager* — *view* to browse, ... — *edit* to change.
Importing needs *Loyalty subscriptions* — *import*, separate again.

A subscription gives a customer an ongoing benefit for a period, rather than a single-use code or amount.

This is the smallest area in Marketing: browse, create, update, expire. There are no bulk operations here — changes are per subscription, or via an import.

Finding a subscription

Search either way round:

- **By customer** — everything that customer has.
- **Most recent** — browse the latest subscriptions across the brand, for a general view of what has been issued.

Searching by customer is the usual path when someone gets in touch.

Creating one

Create the subscription against the customer with its period. For many customers at once, import a phone list instead — see [Importing from a CSV](#).

Updating

An existing subscription can be updated — typically its dates. Use this to extend a subscription, or to correct one entered wrongly.

Expiring

Expiring sets the subscription to end **today**. It stops the benefit from now on without removing the record of it having existed.

There is no delete here, and that is the right design: a subscription a customer has held is part of their history. Expiring is the complete answer for "this should stop".

“ Because expiry lands on today rather than a date you choose, it takes effect immediately. If the customer should keep the benefit until the end of a paid period, **update the end date** instead of expiring.

Common situations

Situation	What to do
Customer says their subscription ended early	Check the end date; update it if it was entered wrongly
Customer wants to cancel	Expire it, or set the end date to the end of their paid period
Subscription created against the wrong customer	Expire it and create the right one. There is no delete
Bulk-issuing subscriptions	Import a phone list — Importing from a CSV

Choosing between a subscription and the alternatives

	Subscription	Personal promotion	Wallet credit
Shape	Ongoing benefit for a period	A specific offer, granted once	A spendable balance
Ends by	Date	Use, or date	Being spent, or date

If the benefit is "free delivery for three months", that is a subscription. If it is "20% off their next order", that is a personal promotion — see [Personal promotions](#).

Related pages

- [Importing from a CSV](#) — issuing subscriptions in bulk
- [Personal promotions](#) — single-use granted offers
- [Wallets: managing records](#) — balance-shaped benefits

Wallets: finding a customer

“ You need: *Wallet manager* — [view](#).

Most wallet work starts with a customer on the phone. This page is how you get from what they can tell you to what you need on screen.

Start from the phone number

Customers know their phone number. They rarely know their customer ID.

The **lookup by phone** resolves a phone number to the customer, which is the identifier the rest of the wallet screens use. Start here.

“ If the lookup finds nobody, the usual cause is a format mismatch — country prefix, spaces — rather than the customer not existing. Try the variant the brand stores before concluding they have no account. The same trap catches CSV imports, see [Importing from a CSV](#).

Check the balance

With the customer identified, the balance answers the first question: how much credit do they have right now.

The balance is the net result of their wallet records. When it doesn't match what the customer expects, the records explain why — some may have expired, some may already be spent.

Read the records

Wallet records are the individual movements making up the balance. Searching them, filtered to that customer, shows how the balance was arrived at.

This is where "I had €20 and now I have €5" gets answered: either the credit was spent, or a record reached its end date.

Open the order behind a record

Where a record is linked to an order, you can open that order's full detail.

This closes the loop on the most common dispute — "I was charged even though I had credit". The order detail shows what was actually applied.

A working order for a customer call

1. **Phone → customer.** Confirm you have the right person.
2. **Balance.** What do they have now?
3. **Records.** How did it get that way?
4. **Order detail**, where a record points at one. What was actually applied?

By step 4 you can usually explain the discrepancy without changing anything — which is the better outcome, since adding credit to close a conversation hides the underlying cause.

When the answer is that something is wrong

If the records genuinely show a mistake, fixing it is on [Wallets: managing records](#) — creating a correcting record, adjusting one, or expiring it.

“ Prefer a **new correcting record** over editing history. It leaves the original visible alongside the correction, which is what anyone reviewing this later needs to see.

Related pages

- [Wallets: managing records](#) — creating and adjusting credit
- [How bulk changes work](#) — before touching many records
- [Vouchers](#) — for a fixed-amount gesture instead of a balance

Wallets: managing records

“ **You need:** *Wallet manager* — *edit* to create or change records. Browsing needs only the view permission. Importing credit needs *Wallet codes* — *import*, separate again.

A wallet record is one movement of credit. The customer's balance is the net of their records — so you never edit a balance directly, you add or adjust the records behind it.

Adding credit

Create a record against the customer with its amount and the dates it is valid between. This is the manual path: service recovery, a goodwill gesture, a correction.

For many customers at once, import a phone list — see [Importing from a CSV](#).

“ **Set an end date deliberately.** Credit with an unnecessarily long life is a liability that sits on the books; credit that expires sooner than the customer expects becomes a complaint. Pick the date on purpose rather than accepting a default.

Adjusting a record

An existing record's **end date and amount** can be updated.

Use this to correct a genuine data-entry error. For anything a customer has already seen, prefer adding a correcting record instead: editing the original rewrites what the customer was told, and leaves nobody able to see what happened.

Expiring records

Expiring ends one or more records immediately. The credit stops being spendable from now on, and the record stays visible.

This is the right tool for credit issued in error, or for closing out a promotion. It is gentler than deletion and it preserves the explanation.

Bulk changes

Wallets follow the standard pattern — read [How bulk changes work](#) first.

The criteria are **starting date, ending date, amount and, optionally, a comment**.

Operation	Effect
Check	Counts non-redeemed records matching the criteria
Bulk update	Updates every matching non-redeemed record
Bulk delete	Removes every matching non-redeemed record

“ **The comment is the most useful criterion you have.** If you tag a batch with a distinctive comment when you create it, you can later act on exactly that batch and nothing else. Without it you are matching on dates and amount, which will eventually catch records from a different campaign that happen to share them.

Credit already spent is never affected, whatever you do here.

Deciding what to do

Situation	Action
Customer owed compensation	Create a record
Credit issued at the wrong amount	Create a correcting record; don't edit the original
Credit issued to the wrong customer	Expire it, then create it on the right customer
A whole batch issued with the wrong dates	Bulk update, matched on the batch's comment
A batch that should never have gone out	Bulk delete — removes only what nobody has spent

Before any bulk operation

1. Run the check and read the count.
2. If it isn't what you expected, your criteria are wrong — especially if you are matching without a comment.
3. Apply, then track to completion.
4. Spot-check a customer from the batch on [Wallets: finding a customer](#).

Related pages

- [Wallets: finding a customer](#) — lookup, balance and records
- [How bulk changes work](#) — the shared pattern
- [Importing from a CSV](#) — issuing credit in bulk
- [Vouchers](#) — a fixed single-use amount instead of a balance