Pricesystem Documentation

Complete PDF Manual

Guide for Magento administrators, technical project owners, and implementation teams.

LAST UPDATED
2026-10-13

SOURCE
Open complete documentation online

EXTENSION
Magento 2 Extension Pricesystem

Table of Contents

Pricesystem Documentation · pricesystem

Overview
Pricesystem
Getting Started
Installation
Configuration
Features
How Priority Works
Quantity-Based Pricing
Cart Price Rule Conditions
Modules
Customer Price
Price Lists
Category Price
Product-Customer Matrix
Price Selection
Global Strategy
Sort Order
Customer Group Overrides
Customer Overrides
Formula Pricing
Price Adjustment
How It Works
Run the Queue Consumer
Troubleshooting
My Prices
Addons
Advanced Config
CSV Import/Export
REST and SOAP APIs
GraphQL
Hyvä Support
Sample Data
Reference

Configuration Fields
Database Reference
Troubleshooting
Common Issues
FAQ

Pricesystem

Price System is the shared pricing foundation behind SoftwareSilo Customer Price, Price Lists, Category Price and Product-Customer Matrix. It decides which available price applies, displays the result in the storefront and provides the common tools used by all four products.

You do not need to buy Price System separately when you purchase one of those products. The required core package is included in that product's installation.

Choose the right pricing product

What you need to manage	Start here
A negotiated product price for one customer	Customer Price
Named price lists shared by customers or groups	Price Lists
A price for every matching product in a category	Category Price
Rule-based prices for products and customer segments	Product-Customer Matrix

You can combine the products. For example, a customer may have a direct Customer Price and also qualify for a Price List. Price System then applies your configured selection strategy.

Shared tools

- Configure Price System
- Understand price selection
- Show customers their prices
- Adjust stored prices in bulk
- Diagnose a calculated price
- Review pricing concepts and common questions

Optional packages

- Advanced Config adds customer and group overrides, custom sort orders and formulas.
- GraphQL exposes calculated prices to GraphQL clients.
- REST and SOAP APIs provide calculated-price and price-record endpoints.
- CSV Import/Export manages larger sets of stored prices.
- Hyvä support adapts storefront price updates to Hyvä.
- Sample Data is intended for development and demonstrations.

Installation

Price System is installed automatically as a dependency of Customer Price, Price Lists, Category Price or Product-Customer Matrix. Install the product package listed in your SoftwareSilo order instead of requiring every Price System package by hand.

If you are installing the complete suite, use the Composer package list supplied with that order. Composer will resolve the shared `mageb2b/pricesystem` package once.

Finish the Magento installation

Run the standard Magento commands after Composer has installed the packages:

```
php bin/magento module:enable MageB2B_ExtensionManager
MageB2B_PricesystemCore
php bin/magento setup:upgrade
php bin/magento cache:clean
```

Enable the modules for the pricing products and add-ons included in the same installation before running `setup:upgrade`.

For production mode, rebuild generated code and static content as required by your deployment process.

Storefront themes

The included frontend layouts cover the standard Magento (Luma) theme and Breeze-based themes. A Hyvä storefront needs the additional `mageb2b/pricesystem-hyva` compatibility package — see Hyvä support.

Check the result

1. Run `php bin/magento module:status MageB2B_PricesystemCore`.
2. Open **Stores > Configuration > MageB2B > Pricesystem**.
3. Confirm **Activate pricesystem for this website** is enabled for the website you are testing.
4. Open the dedicated guide for your pricing product and create one small test record.

Keep the shared core, pricing products and add-ons on compatible release lines. Update the packages from one delivery together instead of upgrading the shared core in isolation.

Configuration

Open **Stores > Configuration > MageB2B > Pricesystem**. Start with the website scope that owns your catalog and customers. Most shops only need the settings below; the remaining fields are useful for unusual storefront or integration setups.

General settings

- **Activate pricesystem for this website** turns Price System calculation on for the selected website.
- **Hide price filter from layered navigation** avoids misleading ranges when Magento's indexed catalog prices differ from customer-specific prices.
- **Reload prices with AJAX for guests** is useful when the NOT LOGGED IN group receives dynamic prices.
- **AJAX price chunk size** limits how many price boxes are refreshed in one request. Leave it at the default unless a large category page needs tuning.
- **Price Cache Lifetime** controls the dedicated calculated-price cache. The value is in **seconds** and defaults to 30 — raise it when prices are stable, lower it when prices change frequently.

Calculation

Custom original price chooses the reference price used by relative calculations and strike-through display. The normal default is the Magento original price.

The following safeguards are enabled by default:

- zero or negative candidates are ignored;
- a Magento tier price is ignored when it is not better than the selected final price.

You can also apply customer or customer-group discounts to selected price types. If both exist, **First Priority** decides which additional discount is considered first.

Where customer and group discounts are entered

The discount values themselves are not entered on this configuration page:

- **Per customer:** open the customer in **Customers > All Customers** and use the **Pricesystem** section. Enter a **Price** value and choose a **Price Application** type.
- **Per customer group:** open the group in **Customers > Customer Groups**. The **Group Price** field and its **Price Application** type appear on the group form.

These assignments also work as cart price rule conditions, so a promotion can require an existing customer or group discount — see Cart Price Rule Conditions.

The Advanced Config add-on extends the same Pricesystem section on the customer form with selection-rule and formula controls — see Customer Overrides.

Display

Enable **Show my prices under my account** when customers should have a searchable list of their available products and prices. You can restrict that page to selected price types or hide it when no custom price exists.

Other display settings control tier prices, strike-through values, discount labels, configurable-product updates and cart labels. Test these options in the storefront used by your shop; themes can place price boxes differently.

Show pricesystem discount on and **Show pricesystem discount type on** choose where the applied discount amount and the winning price type appear: Admin, Frontend, Order email, Cart or Checkout. The exact placement per area is listed in Configuration Fields.

Final-price selection

Choose one global strategy under **Price select rule settings**:

- **Lowest** chooses the lowest valid candidate.
- **Highest** chooses the highest valid candidate.
- **Sort order** chooses the first valid price type in your priority list.
- **Individual price formula** appears when Advanced Config is installed.

See Price Selection before changing this on a live catalog.

Integration field mapping

The API section can map external customer, group, product and category identifiers to Magento attributes. Leave **Use custom API fields** disabled when integrations send Magento entity IDs or the default identifiers documented by the relevant add-on.

Admin permissions

Price System registers its own ACL resources so admin roles can be restricted under **System > Permissions > User Roles**:

Resource	Controls
`MageB2B_PricesystemCore::main`	The Pricesystem entry in the admin menu
`MageB2B_PricesystemCore::price_adjustment`	The Price Adjustment menu item, grid and jobs

Resource	Controls
`MageB2B_PricesystemCore::config`	**Stores > Configuration > MageB2B > Pricesystem**
`MageB2B_PricesystemCoreAdvancedConfig::config`	The same section when the Advanced Config add-on is installed

A role without `::config` cannot open or save the configuration page. Users who only run price adjustments need `::main` and `::price_adjustment`, not the configuration resource.

Logging and cache

Enable the Price System log only while investigating a calculation. The log is written to `var/log/pricesystem.log`.

Price System registers the cache type `pricesystem_cache`:

```
php bin/magento cache:clean pricesystem_cache
```

For targeted diagnosis and warm-up commands, see Troubleshooting.

How Priority Works

Price System can receive several valid prices for the same product, customer and quantity. The global price-selection strategy decides which candidate becomes the final price.

Global strategies

- **Lowest** is common when every candidate represents an offer and the customer should receive the best price.
- **Highest** is useful when the candidates represent minimum contractual values or controlled surcharges.
- **Sort order** gives each installed price type a fixed business priority. The first available type wins.
- **Formula** calculates a result from selected price types and requires Advanced Config.

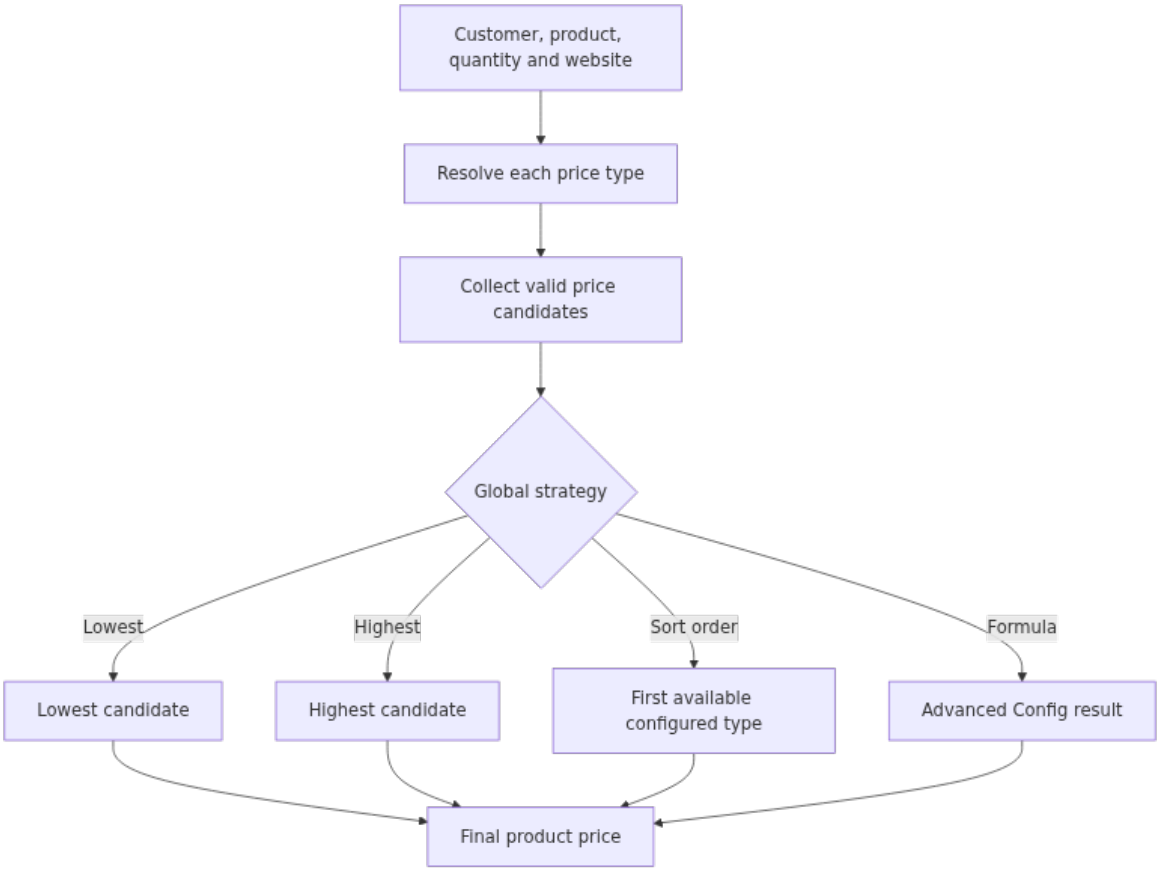
A numeric priority inside Category Price, Price Lists or Product-Customer Matrix resolves competing records of that same price type. It does not replace the global strategy used between different price types.

A practical example

Suppose a product has:

- a Customer Price of €82;
- a Price List value of €79;
- a Category Price of €85.

Lowest returns €79. **Highest** returns €85. With **Sort order**, the result depends on whether Customer Price, Price List or Category Price appears first in the configured list.



When a result is unexpected, inspect the price type's own assignment rules first, then check the global strategy.

Quantity-Based Pricing

Customer Price, Price Lists, Category Price and Product-Customer Matrix can store more than one price for the same assignment by using different minimum quantities.

For an order quantity of 18, a typical set of breaks at 1, 10 and 25 uses the row for 10. The row for 25 does not apply yet.

Set up sensible breaks

1. Create a quantity-1 row so the assignment has a base value.
2. Add each higher minimum quantity as a separate row.
3. Keep website and validity dates consistent unless the tiers intentionally apply in different periods.
4. Test quantities immediately below, exactly on and immediately above every break.

The price value may be fixed or relative, depending on the selected application type. A lower numeric value is not automatically chosen merely because it is cheaper; quantity eligibility is resolved before the global price-selection strategy compares price types.

Price tiers for one product assignment belong in that product's guide:

- Customer Price quantity tiers
- Create Price List product prices
- Create Category Prices
- Create a Product-Customer Matrix

Cart Price Rule Conditions

Price System adds two conditions to Magento cart price rules so a promotion can target customers who already carry a Price System discount.

Open **Marketing > Cart Price Rules**, expand **Conditions** and add a condition. The condition picker contains **Customer Discount** and **Customergroup Discount**:

Condition	Yes when
Customer has customer discount	The quote's customer has a **Price** value and a **Price Application** type in the Pricesystem section of the customer record.
Customer has customergroup discount	The quote's customer group has a **Group Price** value and a **Price Application** type on the group form.

Each condition compares against a Yes/No value. Guests always evaluate to **No**, because the check needs a signed-in customer and a customer group on the quote.

The discount values themselves are maintained on the customer and customer-group forms — see Where customer and group discounts are entered. A cart price rule applies on top of the calculated final price, so test the combined result on a staging store before enabling both on the same products.

Customer Price

Use Customer Price when one customer has a negotiated value for one product. The assignment can include quantity breaks, a website and a validity period.

This is the most direct pricing model in the suite. It does not create a reusable named list or infer products from a category or attribute rule.

Continue with the Customer Price guide.

Price Lists

Use Price Lists when a named set of product prices should be shared by several customers or customer groups. Lists can be assigned directly or matched with customer and address conditions.

A list can have dates and its own internal priority. Product rows can contain quantity breaks. A list marked **Is Base Pricelist** is exposed as the separate **Base Pricelist** price type. It behaves as a fallback only when the Price System selection strategy gives **Pricelist** precedence and checks **Base Pricelist** next.

Continue with the Price Lists guide.

Category Price

Use Category Price when all matching products in a Magento category should receive the same pricing rule for one customer or customer group.

The rule follows product category membership. It does not copy a price row to every product, and it is separate from Product-Customer Matrix conditions.

Continue with the Category Price guide.

Product-Customer Matrix

Use Product-Customer Matrix when a price should apply to products selected by SKU, category or product-attribute conditions and to customers selected by direct assignment or a customer attribute.

Matrices are useful for segment rules such as “all safety products for the Wholesale group”. They are more flexible than a direct Customer Price and more rule-driven than a named Price List.

Continue with the Product-Customer Matrix guide.

Price Selection

Price System first collects every price that is valid for the current product, customer, quantity, website and date. It then uses one selection strategy to choose the final value.

There are two different kinds of priority:

1. A price product resolves its own competing records. A Product-Customer Matrix, for example, can use matrix priority when two matrices match.
2. Price System compares the resulting price types. This is where Lowest, Highest, Sort order or Formula applies.

Keeping those steps separate makes most pricing questions easier to diagnose.

Choose a strategy

- Global system configuration
- Sort order
- Customer-group overrides
- Customer overrides
- Formula pricing

Customer and group overrides, custom sort orders and formulas require Advanced Config.

Global Strategy

Open **Stores > Configuration > MageB2B > Pricesystem > Price select rule settings**.

Choose the rule that should apply when no more specific Advanced Config rule is active:

- **Lowest** returns the lowest valid candidate.
- **Highest** returns the highest valid candidate.
- **Sort order** returns the first available price type in **Price Priority Order**.
- **Individual price formula** calculates a value from selected price types and appears with Advanced Config.

Save the configuration at the correct scope, clean the Price System cache and test one customer whose product has at least two competing prices.

Do not use Lowest merely because it sounds customer-friendly. Relative surcharges, minimum prices or a deliberately authoritative Customer Price can make Sort order the safer business rule.

Sort Order

Sort order is useful when one source should always outrank another, regardless of the numeric values.

In **Price Priority Order**, place the authoritative price type first. Price System walks down the list and uses the first type that has a valid value for the current context.

The list mixes Pricesystem price types (Customer Price, Pricelist, Categoryprice, Product Customer Matrix — one entry per installed price module) with Magento's built-in candidates. The built-in entries use these labels: **Group Price** (this is Magento's *tier price* — do not confuse it with **Group Discount**, the customer-group discount), **Special Price**, **Customer Discount**, **Group Discount**, **Catalog Price Rule** and **Original Price**.

For example:

1. Customer Price
2. Pricelist
3. Product Customer Matrix
4. Categoryprice
5. Catalog Price Rule
6. Original Price

If Customer Price has no valid row for the requested quantity or date, the next available type is considered.

Advanced Config can replace the global order for a customer or customer group. Keep those exceptions rare and document why they exist; otherwise two customers with similar assignments can be difficult to compare.

Customer Group Overrides

Advanced Config adds **Price Select Rule** to the Magento customer-group form.

Use a group override when every customer in that group should follow a different strategy from the storefront default. Available choices include the global configuration, Lowest, Highest, a custom sort order and an individual formula.

A customer-level rule can take precedence over the group. When testing a group change, use a customer without a separate customer override and clean the Price System cache.

For a custom sort order, arrange the price types in the group form. Position 1 is the highest priority.

Customer Overrides

The customer edit form already contains a **Pricesystem** section provided by the base module — it holds the customer-level **Price** and **Price Application** fields. Advanced Config extends that same section with the selection-rule and formula controls described here. Use it for an exceptional contract that cannot follow the customer's normal group strategy.

The customer can use Lowest, Highest, the system configuration, a custom sort order or an individual formula. A saved customer-specific sort order is treated as an explicit override.

Before adding one, check whether a direct Customer Price or a separate Price List would represent the agreement more clearly. A selection override changes how all available price types are compared, not just one product.

After saving, clean the Price System cache and test both a product with several candidates and one with only a catalog price.

Formula Pricing

Formula pricing is part of Advanced Config. It combines available price types arithmetically instead of choosing only one.

A formula can be configured globally, for a customer group or for one customer. Resolution follows the most specific available formula: customer, then group, then global.

Price codes and helpers

Use the price codes available in your installation, such as `customer_price`, `pricelist` or `orig_price`.

Supported helpers include:

- `low(code)`, `high(code)` and `avg(code)` when a price type provides more than one value;
- `min(...)`, `max(...)`, `round(...)` and `abs(...)`.

Example:

```
low(customer_price) * 0.95 + 2
```

The editor validates unknown price codes, unsupported functions and invalid syntax.

Missing values

Choose one of two behaviors:

- stop formula evaluation and use the configured fallback;
- treat a missing input as zero.

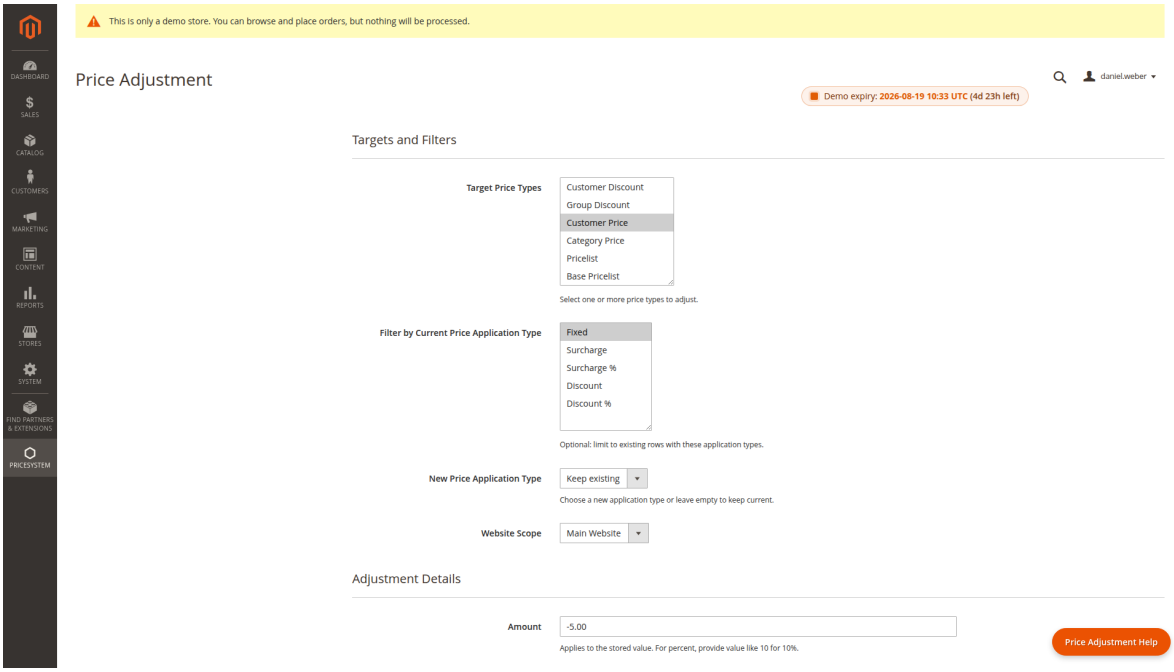
Fallback can use Lowest, Highest, Sort order or Original price. For a formula that refers to an optional price type, fallback is normally safer than silently treating the missing value as zero.

Test the formula with every combination of inputs that can occur in the catalog, including a product with no custom price and a price that has expired.

Price Adjustment

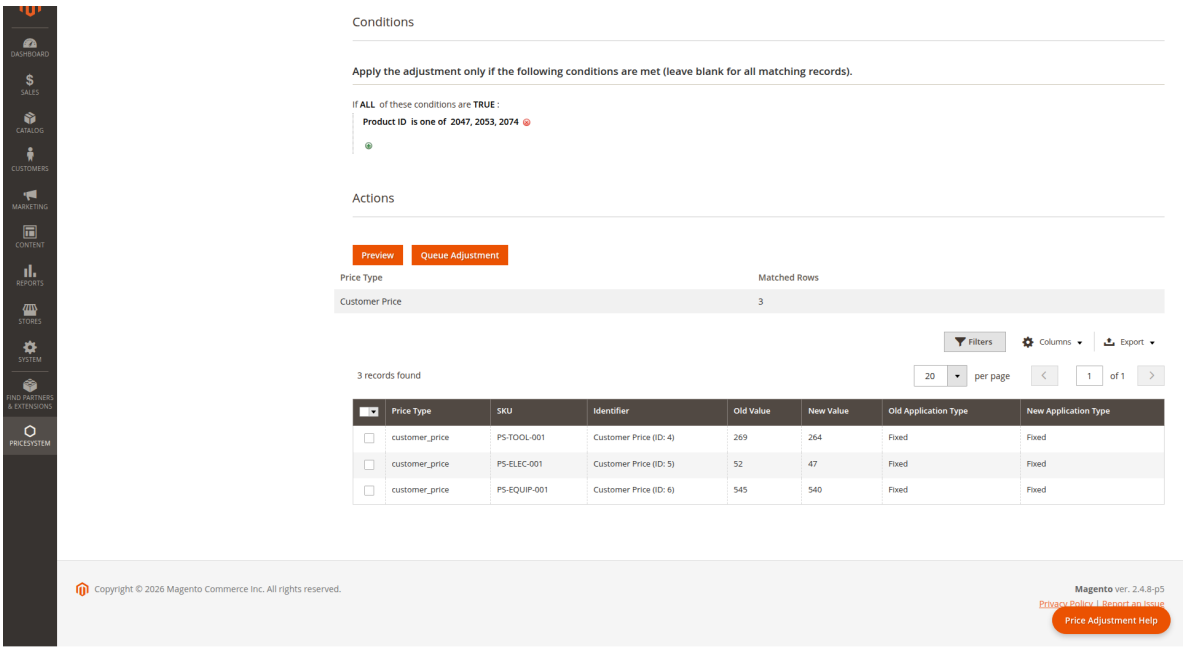
Price Adjustment changes existing stored pricing records in bulk. Use it for controlled tasks such as adding €5 to selected Price List rows or converting a set of fixed Customer Prices to percentage discounts.

It does not edit Magento's catalog price directly.



Safe workflow

1. Open **Pricesystem > Price Adjustment**.
2. Select one or more target price types.
3. Narrow the records by website, current application type and conditions.
4. Enter the amount and, when needed, a new application type or validity dates.
5. Click **Preview**.
6. Review the matched counts and old/new values. Export the preview CSV for a larger change.
7. Click **Queue Adjustment** only when the preview is correct.
8. Confirm the job reaches **Completed** and spot-check calculated storefront prices.



The preview is not the write operation. The queued job performs the actual update through Magento's message queue.

Supported targets

The available list depends on installed packages and can include customer and group discounts, Customer Price, Category Price, Price Lists, Base Price Lists and Product-Customer Matrix.

Use How Price Adjustment Works for application types and date handling, and Run the Queue Consumer before scheduling a production change.

How It Works

Price Adjustment reads the stored rows selected by your target types and conditions, calculates a proposed new value, and builds a preview. The queued consumer repeats the rules against the job and writes the changes.

Amount

The amount is applied to the stored value:

- `5` adds five units when the operation represents a nominal increase;
- `5` represents five percent when the selected operation is percentage-based.

Always read the preview's **Old Value**, **New Value**, **Old Application Type** and **New Application Type** columns instead of inferring the result from the amount alone.

Application type

You can keep the current type or replace it with:

- Fixed
- Surcharge
- Surcharge %
- Discount
- Discount %

Relative types need a meaningful base price. Preview at least one row from every target type because Price Lists, Customer Prices and matrix records may use different base-price settings.

Dates

When the target supports dates, setting both dates can create a dated record rather than overwriting the current one. Use this for a planned price period only after checking the preview for duplicate or overlapping rows.

Conditions

Leaving conditions empty addresses all records matching the selected targets and website. For production work, start with a narrow SKU, customer, group or identifier condition and expand only after the result is understood.

Run the Queue Consumer

Queued adjustments are processed by:

```
mageb2b.pricesystem.price_adjustment.consumer
```

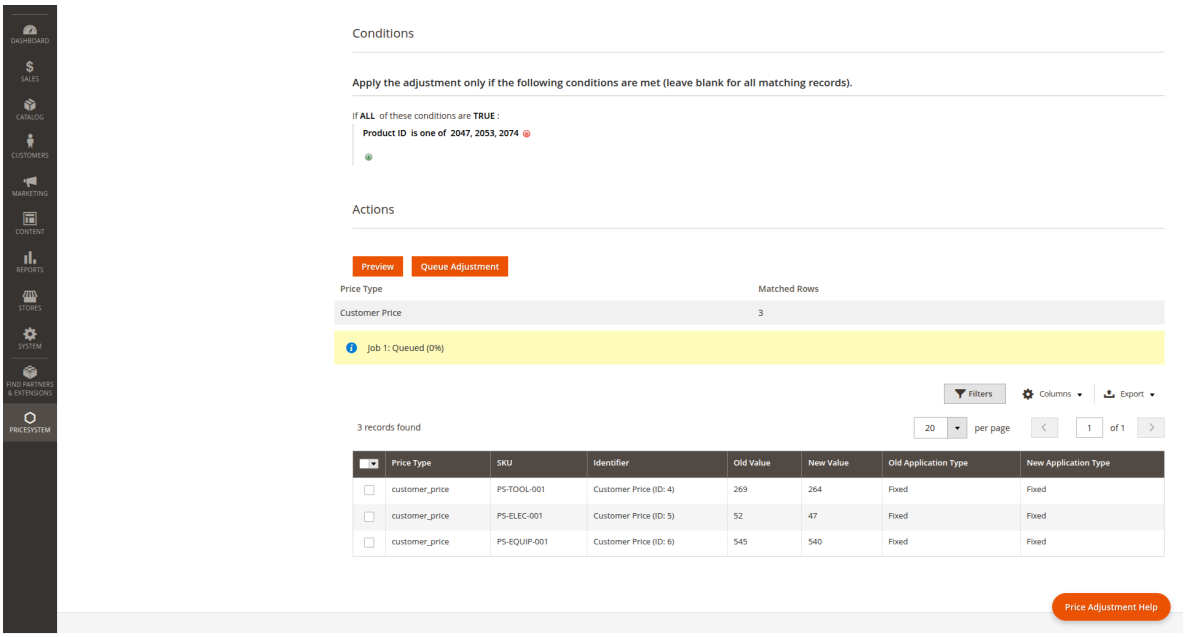
Run it directly for a controlled one-job check:

```
php bin/magento queue:consumers:start \
  mageb2b.pricesystem.price_adjustment.consumer \
  --max-messages=1
```

For normal operation, add the consumer to the queue worker or process supervisor used by the shop. Do not rely on an administrator keeping a terminal open.

Job states

- **Queued**: published and waiting for the consumer.
- **Processing**: the consumer has started the job.
- **Completed**: the selected records were processed.
- **Failed**: the job stopped and its error must be reviewed.



After completion, clean or warm the Price System cache according to the size of the change and verify a representative customer/SKU calculation.

Troubleshooting

The job stays queued

- Confirm `mageb2b.pricesystem.price_adjustment.consumer` is running.
- Check Magento's queue connection and failed-message handling.
- Start the consumer with `--max-messages=1` and watch its output.
- Review Magento logs for the job ID.

Preview matches too many rows

Return to the form and add a website, application-type or rule condition. Do not queue the job and plan to repair unwanted records afterward.

The preview grid shows a limited sample while the summary can represent a larger total. Export the preview CSV when the sample is not enough for review.

Preview is empty

Check that:

- the selected price product is installed;
- records exist at the selected website;
- the current application-type filter matches those records;
- date and rule conditions apply;
- identifiers such as SKU, customer or group refer to the intended entities.

A job failed

Record the job ID and error before retrying. Fix the underlying data or queue problem, create a fresh preview and submit a new job. Do not alter the failed job record in the database to make it appear completed.

Values changed but the storefront did not

Clean `pricesystem_cache` and the relevant Magento page caches, then calculate one affected SKU with `pricesystem:get-final-price`. This separates a correct stored update from a stale storefront response.

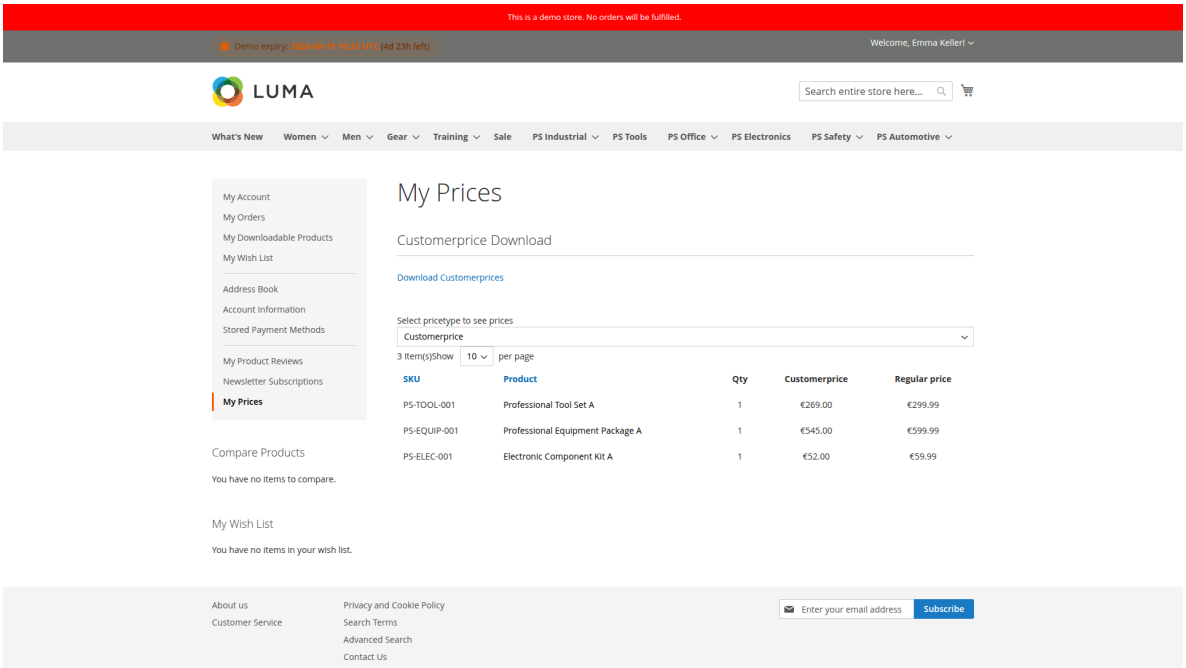
My Prices

My Prices gives a signed-in customer one account page for products that currently have an applicable custom price.

Enable it under **Stores > Configuration > MageB2B > Pricesystem > Display settings**.

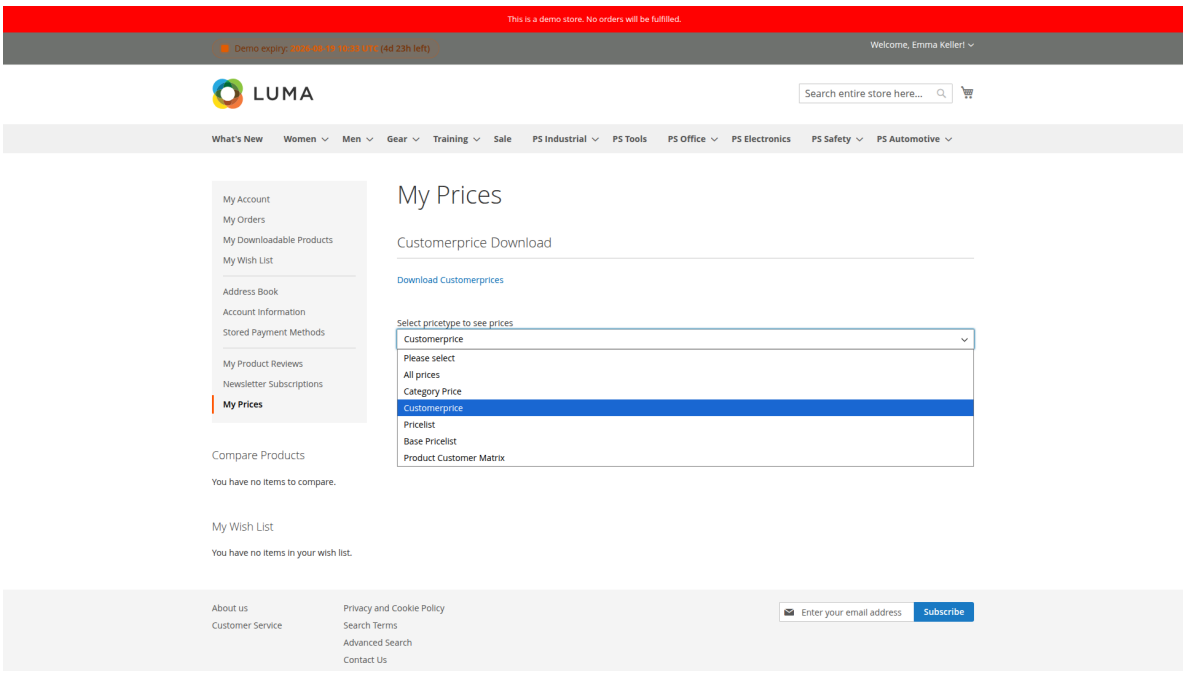
What customers see

The table can show the product, SKU, quantity break, custom value and regular price. The exact rows depend on the installed price products and the current website and date.



Use **Price types shown in “My Prices” dropdown** to limit the selector. Enable **Show my prices only when customer has custom prices assigned** if the navigation item should disappear for customers without relevant records.

The selector also has an **All prices** entry that merges the records of every installed price type into one combined view instead of filtering by a single type. It appears automatically when the dropdown configuration is left empty; otherwise select it explicitly in the multi-select.



CSV links

Customer Price, Price Lists, Category Price and Product-Customer Matrix can add their own CSV download links. Each product controls its download setting and delimiter separately.

A link is hidden when that product has no exportable row for the current customer, website and date.

```
name;sku;qty;price
"Electronic Component Kit A";PS-ELEC-001;1;€52.00
"Professional Equipment Package A";PS-EQUIP-001;1;€545.00
"Professional Tool Set A";PS-TOOL-001;1;€269.00
```

If a product is missing

1. Confirm the product is enabled and assigned to the current website.
2. Check the customer or group assignment.
3. Check quantity and validity dates.
4. Confirm that the price type is allowed in the My Prices dropdown.
5. Clean the Price System cache.

Advanced Config

Advanced Config is the shared add-on for businesses that need different price-selection rules by customer or customer group.

What it adds

- Lowest, Highest, system configuration, custom sort order or formula at customer and group level;
- an individual price-type order for a customer or group;
- global, group and customer formulas;
- a **Discountable** product flag;
- a **Verified** customer flag;
- optional checks that restrict selected price types to discountable products or verified customers.

Where to work

- Global defaults and formula behavior: **Stores > Configuration > MageB2B > Pricesystem**
- Customer rule: **Customers > All Customers**, then the **Pricesystem** section
- Group rule: **Customers > Customer Groups**

A direct customer rule is the most specific. Group and global settings supply the less specific defaults.

Configuration conflict warning

Saving **Stores > Configuration > MageB2B > Pricesystem** shows a warning when **Select pricetypes for which discountable check will be applied** contains the same price type configured as **Custom original price**. For products marked as not discountable, that reference price then cannot be calculated and the custom-price column in backend grids shows `N/A`. Remove the overlapping price type or choose a different custom original price.

Formula safeguards

The formula editor accepts installed price codes and a limited set of arithmetic helpers. Missing inputs can either trigger a fallback strategy or be treated as zero. See Formula Pricing before enabling formulas for production customers.

Installation

Install `mageb2b/pricesystem-advancedconfig` from the repository configured for

your SoftwareSilo order, then run Magento's normal upgrade and cache commands. Composer also keeps the shared Price System dependency compatible.

CSV Import/Export

CSV Import/Export is installed per pricing product. It adds Magento Import/Export entities and matching command-line tools for bulk maintenance.

This add-on is different from the customer-facing CSV links on **My Prices**. Those links are part of the base pricing products.

Packages and entities

Pricing product	Package	Magento import entity
Customer Price	`mageb2b/pricesystem-customerprice-importexport`	`pricesystem_customerprice`
Price Lists	`mageb2b/pricesystem-pricelist-importexport`	`pricesystem_pricelist`
Category Price	`mageb2b/pricesystem-categoryprice-importexport`	`pricesystem_categoryprice` and `pricesystem_categoryprice_customergroup`
Product-Customer Matrix	`mageb2b/pricesystem-productcustomermatrix-importexport`	`pricesystem_product_customer_matrix`

Use **System > Data Transfer > Export** first and use that file as the template for import. This is safer than composing column names from memory and preserves the identifiers expected by the installed packages.

Command-line tools

```
php bin/magento pricesystem:export-customerprice
php bin/magento pricesystem:import-customerprice
path/to/file.csv

php bin/magento pricesystem:export-pricelist
php bin/magento pricesystem:import-pricelist path/to/file.csv

php bin/magento pricesystem:export-categoryprice
php bin/magento pricesystem:import-categoryprice
path/to/file.csv
php bin/magento pricesystem:export-categoryprice-customergroup
php bin/magento pricesystem:import-categoryprice-customergroup
path/to/file.csv

php bin/magento pricesystem:export-productcustomermatrix
php bin/magento pricesystem:import-productcustomermatrix
path/to/file.csv
```

Run `php bin/magento help COMMAND` for the options supported by a particular importer or exporter.

Safe import routine

1. Export the current records.
2. Keep a recoverable backup outside Magento's import directory.
3. Test the modified file in a staging environment.
4. Use Magento's validation step before import.
5. Import a small batch and verify calculated storefront prices.
6. Clean the Price System cache after the final batch.

The dedicated product guides explain their customer, SKU and other aliases.

REST and SOAP APIs

Price System Core provides calculated-price endpoints. Optional API packages add CRUD routes for the stored records owned by Customer Price, Price Lists, Category Price and Product-Customer Matrix.

Magento exposes the same service contracts through its normal REST and SOAP layers.

Calculated-price routes

Context	One SKU	Several SKUs
Integration or authorized admin	`POST /V1/pricesystem/get-final-price`	`POST /V1/pricesystem/get-multi-price`
Signed-in customer	`POST /V1/pricesystem/me/get-final-price`	`POST /V1/pricesystem/me/get-multi-price`
Current storefront context	`POST /V1/pricesystem/context/get-final-price`	`POST /V1/pricesystem/context/get-multi-price`

REST requests normally use `/rest/V1/...` or `/rest/{storeCode}/V1/...`.

The `me` routes require a customer token and derive the customer from that token. The context routes use the current signed-in customer or the NOT LOGGED IN group. Do not send an arbitrary customer ID to either of those contexts.

Record APIs

Install only the package for the product an integration must manage:

- `mageb2b/pricesystem-customerprice-api`
- `mageb2b/pricesystem-pricelist-api`
- `mageb2b/pricesystem-categoryprice-api`
- `mageb2b/pricesystem-productcustomermatrix-api`

Each product guide lists its exact resources and routes. Use a Magento integration with only the ACL resources it needs rather than an administrator token embedded in application code.

Order item extension attributes

Order reads such as `GET /V1/orders/:id` expose two Price System extension attributes on each item:

Attribute	Type	Content

Attribute	Type	Content
`ps_discount_amount`	float	Discount amount Price System recorded on the item
`ps_applied_pricetype`	string	Code of the price type that supplied the final price

The values are stored on the quote item during totals collection, copied to the order item and returned inside `extension_attributes`. They are read-only snapshots of the applied pricing, useful for order exports and ERP integrations.

The related customer attributes `pricesystem_price` and `pricesystem_price_type` are protected: writing them through `custom_attributes` is accepted only from admin and integration tokens, while customer and guest tokens are rejected. See Protected customer attributes for the shared mechanism.

After a write

Successful repository writes invalidate relevant Price System calculations. For a bulk integration, still verify one affected customer and SKU after the batch and monitor failed API responses before treating the synchronization as complete.

GraphQL

Install `mageb2b/pricesystem-graph-ql` when a headless storefront or integration needs calculated Price System values through Magento GraphQL.

Queries

Query	Context
`pricesystemFinalPrice`	Explicit customer, group, quantity, date and website input
`pricesystemMultiSkuPrice`	The same explicit context for several SKUs
`pricesystemMyFinalPrice`	Authenticated customer's context
`pricesystemMyMultiSkuPrice`	Authenticated customer's context for several SKUs
`pricesystemContextFinalPrice`	Current storefront context
`pricesystemContextMultiSkuPrice`	Current storefront context for several SKUs

Multi-SKU results include the SKU, final price and applicable quantity tiers.

Use the `My` queries with a customer token. Use the context queries when the storefront session should determine the customer or guest group. Explicit customer or group inputs should be protected as integration data and must not be exposed as an unrestricted public price lookup.

The Advanced Config add-on has its own GraphQL package, `mageb2b/pricesystem-advancedconfig-graph-ql`. Each pricing product also has its own optional GraphQL package for reading stored records.

Advanced Config query

`pricesystemAdvancedConfig(customerId, groupId)` returns the price-selection configuration that applies to a customer or customer group. Both arguments are optional; pass the IDs you want to inspect.

The result combines the raw configured values at each level with the effective values after precedence resolution:

Field group	Fields	Content
Global formula	`formula`, `missing_input_behavior`, `fallback_strategy`	System-level formula and its failure handling

Field group	Fields	Content
System	`system_price_select_rule`, `system_price_sort_order`	Global rule and price-type order
Customer	`customer_id`, `customer_priceselect`, `customer_price_formula`, `customer_price_sort_order`	Raw values stored on the customer
Group	`group_id`, `group_priceselect`, `group_price_formula`, `group_price_sort_order`	Raw values stored on the group
Resolved	`resolved_price_select_rule`, `resolved_price_select_source`, `resolved_formula`, `resolved_formula_source`, `resolved_price_sort_order`, `resolved_price_sort_order_source`	Effective values and which level supplied them

Mind the numbering: `customer_priceselect` and `group_priceselect` use the customer-form rule values, while `resolved_price_select_rule` is translated to the system rule (`1` Lowest, `2` Highest, `3` Sort order, `4` Individual price formula). Each `resolved_*_source` is `customer`, `group` or `system` — a direct customer rule wins over the group rule, which wins over the system default. `resolved_price_sort_order` is only populated when the effective rule is Sort order; sort orders are returned as lists of price-type codes.

```
query AdvancedConfig($customerId: Int, $groupId: Int) {
  pricesystemAdvancedConfig(customerId: $customerId, groupId:
$groupId) {
    formula
    missing_input_behavior
    fallback_strategy
    system_price_select_rule
    system_price_sort_order
    customer_id
    customer_priceselect
    customer_price_formula
    customer_price_sort_order
    group_id
    group_priceselect
    group_price_formula
    group_price_sort_order
    resolved_price_select_rule
    resolved_price_select_source
    resolved_formula
```

```
    resolved_formula_source
    resolved_price_sort_order
    resolved_price_sort_order_source
  }
}
```

Treat this query as integration data: it accepts arbitrary customer and group IDs and exposes each target's effective pricing configuration, so it should not be reachable as an unrestricted public lookup.

Hyvä Support

Install `mageb2b/pricesystem-hyva` when the storefront uses a Hyvä theme.

Prerequisites

- A working Hyvä theme (`hyva-themes/magento2-default-theme` or a theme based on it).
- The Hyvä compatibility-module fallback (`Hyva_CompatModuleFallback`, package `hyva-themes/magento2-compat-module-fallback`). The Price System Hyvä module registers itself in the compat-module registry and lists the module in its load sequence, but the Composer package is not required by `composer.json` — install it explicitly if your Hyvä setup does not already include it.
- Include the module's Tailwind sources under `view/frontend/tailwind/` in your theme's Tailwind build, the same way other Hyvä compatibility modules are merged, so its utility classes and templates are covered by the generated stylesheet.

What it adapts

- price refreshes on product and category pages;
- configurable-product option changes;
- quantity-tier display;
- My Prices;
- discount information in the cart.

Composer resolves the compatible Price System and Hyvä theme packages. After installation, run the normal Magento upgrade, generated-code and cache steps used by your deployment.

Test a category page, a configurable product, quantity changes, the cart and My Prices with a customer who has real pricing assignments. No separate Hyvä configuration is required for ordinary setups.

Breeze storefronts

Breeze-based themes do not need this package. The core Price System module already ships a Breeze compatibility layout that enables its dynamic price reload on Breeze storefronts — the `pricesystem-hyva` package is only needed for Hyvä.

Sample Data

The sample-data packages create a small pricing catalog for demonstrations and development. Do not install them in a production shop.

The shared package `mageb2b/pricesystem-sample-data` creates the products, categories, customer groups, customers and addresses used by the examples. A product-specific sample package then adds only its own price records.

Product packages

- `mageb2b/pricesystem-customerprice-sample-data`
- `mageb2b/pricesystem-pricelist-sample-data`
- `mageb2b/pricesystem-categoryprice-sample-data`
- `mageb2b/pricesystem-productcustomermatrix-sample-data`

Install the shared package and the sample package for the pricing product you want to demonstrate, then run `php bin/magento setup:upgrade`.

Verify and reset

Each package provides commands in these forms:

```
php bin/magento sampledata:verify:pricesystem
php bin/magento sampledata:reset:pricesystem

php bin/magento sampledata:verify:pricesystem-customerprice
php bin/magento sampledata:reset:pricesystem-customerprice
```

Replace the suffix with `pricelist`, `categoryprice` or `productcustomermatrix` for the other products.

Reset commands ask for confirmation and remove the matching sample records. A product-specific reset leaves the shared products and customers in place.

Configuration Fields

This reference lists the Magento Admin settings exposed by the base package and its documented add-ons: `mageb2b/pricesystem`, `mageb2b/pricesystem-advancedconfig`.

Configuration paths are included for controlled deployments and `config:set`. Use Magento Admin for normal changes so field validation and scope inheritance still apply. The extension status display is not listed because it reports installation state rather than a configurable value.

`mageb2b/pricesystem`

General settings

Field	Configuration path	Input	Scope
Activate pricesystem for this website	`pricesystem/general/active`	Select	Default, Website
Hide price filter from layered navigation for customers	`pricesystem/general/hide_price_filter`	Select	Default, Website, Store view
Reload prices with ajax for guests	`pricesystem/general/reload_price_for_guest`	Select	Default
Routes to skip pricesystem ajax price reload	`pricesystem/general/skip_ajax_price_routes`	Text area	Default
Use native JSON encode	`pricesystem/general/use_native_json_encode`	Select	Default, Website, Store view
AJAX price chunk size	`pricesystem/general/ajax_chunk_size`	Text	Default, Website, Store view
Price Cache Lifetime	`pricesystem/general/price_cache_lifetime`	Text	Default

Price calculation settings

Field	Configuration path	Input	Scope
Custom original price	`pricesystem/price_calculation/custom_original_price`	Select	Default
Skip greater tier prices	`pricesystem/price_calculation/skip_greater_tier_price`	Select	Default, Website, Store view
Disable pricesystem for selected prices	`pricesystem/price_calculation/disabled_prices`	Multi-select	Default
Skip Zero Price	`pricesystem/price_calculation/skip_zero_price`	Select	Default, Website, Store view
Apply customer discount on selected prices	`pricesystem/price_calculation/apply_customer_discount_on`	Multi-select	Default
Apply group discount on selected prices	`pricesystem/price_calculation/apply_group_discount_on`	Multi-select	Default
First Priority	`pricesystem/price_calculation/first_priority_customer_discount_or_group`	Select	Default

Display settings

Field	Configuration path	Input	Scope
Show my prices under my account	`pricesystem/display/custom_prices_my_account`	Select	Default, Website, Store view
Price types shown in "My Prices" dropdown	`pricesystem/display/custom_prices_my_account_provider_types`	Multi-select	Default, Website, Store view
Show my prices only when customer has custom prices assigned	`pricesystem/display/custom_prices_my_account_not_empty`	Select	Default, Website, Store view
Standard price strikethrough	`pricesystem/display/standard_price_strikethrough`	Select	Default, Website, Store view
Show custom price column in backend grids (customer price grid, pricelist products grid)	`pricesystem/display/show_custom_price_column`	Select	Default
Show tierprices	`pricesystem/display/show_tierprices`	Select	Default, Website, Store view

Field	Configuration path	Input	Scope
Specify css class identifier to get all price box on product list page.	`pricesystem/display/css_class_for_product_list_price_element`	Text	Default, Website, Store view
Specify css classes to get all additional elements which needs to be updated with price on product list page.	`pricesystem/display/css_class_for_product_list_price_additional_element`	Text	Default, Website, Store view
Show tierprices on list page	`pricesystem/display/show_tierprices_on_listpage`	Select	Default, Website, Store view
Enable configurable option prices on listing page	`pricesystem/display/enable_configurable_update_on_list`	Select	Default, Website, Store view
Display discount text on product view pages	`pricesystem/display/frontend_show_discount_percent_product_view`	Select	Default, Website, Store view
Display discount text on listing pages	`pricesystem/display/frontend_show_discount_percent_product_list`	Select	Default, Website, Store view
Discount Text	`pricesystem/display/frontend_discount_text`	Text	Default, Website, Store view
Discount Text for configurable product used on list page	`pricesystem/display/frontend_conf_product_discount_text_listpage`	Text	Default, Website, Store view
Discount text css style	`pricesystem/display/frontend_discount_text_style`	Text	Default, Website, Store view
Hide main prices when min price block is there on list page	`pricesystem/display/hide_main_price_when_minimalprice_price_exist`	Select	Default, Website, Store view
Show discount text on cart	`pricesystem/display/show_discount_text_on_cart`	Select	Default, Website, Store view
Discount text to be used on cart	`pricesystem/display/discount_text_for_cart`	Text area	Default, Website, Store view
Discount text style to be used on cart	`pricesystem/display/discount_text_style_for_cart`	Text area	Default, Website, Store view
Round up discount percent for display?	`pricesystem/display/roundup_discount_percent_for_display`	Select	Default, Website

Field	Configuration path	Input	Scope
Show pricesystem discount on	`pricesystem/display/show_discount_on`	Multi-select	Default, Website
Show pricesystem discount type on	`pricesystem/display/show_discount_type_on`	Multi-select	Default, Website
Hide price on list page for customer groups	`pricesystem/display/hide_price_on_list_page_customer_group`	Multi-select	Default, Website
Show From Price for Configurables	`pricesystem/display/show_price_from`	Select	Default, Website, Store view

Visibility dependencies:

- `pricesystem/display/custom_prices_my_account_provider_types` is available when `pricesystem/display/custom_prices_my_account` is `1`.
- `pricesystem/display/custom_prices_my_account_not_empty` is available when `pricesystem/display/custom_prices_my_account` is `1`.
- `pricesystem/display/discount_text_for_cart` is available when `pricesystem/display/show_discount_text_on_cart` is `1`.
- `pricesystem/display/discount_text_style_for_cart` is available when `pricesystem/display/show_discount_text_on_cart` is `1`.

The multi-selects `pricesystem/display/show_discount_on` and `pricesystem/display/show_discount_type_on` accept these area options:

Option	Where the value appears
Admin	Item grid of the admin order view
Frontend	Order view in the customer account
Order email	Item table of order confirmation emails
Cart	Shopping cart item rows
Checkout	Checkout item data shown during checkout

`show_discount_on` controls the calculated discount amount; `show_discount_type_on` controls the applied price-type label. Both are checked per area, so an integration or a custom layout can read the same flags.

Price select rule settings

Field	Configuration path	Input	Scope
Price Select Rule	`pricesystem/price_select_rule/priceselect`	Select	Default

Field	Configuration path	Input	Scope
Next available price as strike through price	`pricesystem/price_select_rule/use_next_available_price_in_strike_through`	Select	Default
Skip customer and group discounts application on next available price	`pricesystem/price_select_rule/skip_customer_group_discounts_for_next_available_price`	Select	Default
Price Priority Order	`pricesystem/price_select_rule/price_sort_order`	Custom control	Default

Visibility dependencies:

- `pricesystem/price_select_rule/use_next_available_price_in_strike_through` is available when `pricesystem/price_select_rule/priceselect` is `3`.
- `pricesystem/price_select_rule/skip_customer_group_discounts_for_next_available_price` is available when `pricesystem/price_select_rule/priceselect` is `3`.
- `pricesystem/price_select_rule/price_sort_order` is available when `pricesystem/price_select_rule/priceselect` is `3`.

API settings

Field	Configuration path	Input	Scope
Use custom API fields for sync	`pricesystem/api/use_custom_api_fields`	Select	Default
Customer attribute	`pricesystem/api/customer_attribute`	Select	Default
Customer group attribute	`pricesystem/api/customer_group_attribute`	Select	Default
Product attribute	`pricesystem/api/product_attribute`	Select	Default
Category attribute	`pricesystem/api/category_attribute`	Select	Default

Visibility dependencies:

- `pricesystem/api/customer_attribute` is available when `pricesystem/api/use_custom_api_fields` is `1`.
- `pricesystem/api/customer_group_attribute` is available when

`pricesystem/api/use_custom_api_fields` is `1`.

- `pricesystem/api/product_attribute` is available when `pricesystem/api/use_custom_api_fields` is `1`.
- `pricesystem/api/category_attribute` is available when `pricesystem/api/use_custom_api_fields` is `1`.

Debug

Field	Configuration path	Input	Scope
Enable log	`pricesystem/debug/enable_log`	Select	Default

`mageb2b/pricesystem-advancedconfig`

Advanced settings

Field	Configuration path	Input	Scope
Default value for discountable	`pricesystem/advanced/default_discountable`	Select	Default, Website
Select pricetypes for which discountable check will be applied	`pricesystem/advanced/discountable_restricted_pricetypes`	Multi-select	Default, Website
Discountable check for additional Customer/Group Discount	`pricesystem/advanced/force_discountable_discount_check`	Select	Default, Website
Default value for verified	`pricesystem/advanced/default_verified`	Select	Default, Website
Default value for priceselect	`pricesystem/advanced/default_priceselect`	Text	Default, Website
Select pricetypes for which verified check will be applied	`pricesystem/advanced/verified_restricted_pricetypes`	Multi-select	Default, Website

`pricesystem/advanced/default_priceselect` is a raw text field, not a select. It assigns a customer-level price-select rule to new customers, so enter the numeric value used by the customer form: `0` Default, `1` Lowest, `2` Highest, `4` Price select from system config, `5` Custom sort order, `6` Individual price formula. These values differ from the global `pricesystem/price_select_rule/priceselect` codes.

Price select rule settings

Field	Configuration path	Input	Scope
Individual price formula	`pricesystem/price_select_rule/formula`	Custom control	Default
Missing values behavior	`pricesystem/price_select_rule/formula_missing_input_behavior`	Select	Default
Formula error fallback	`pricesystem/price_select_rule/formula_fallback_strategy`	Select	Default

Visibility dependencies:

- `pricesystem/price_select_rule/formula` is available when `pricesystem/price_select_rule/priceselect` is `4`.
- `pricesystem/price_select_rule/formula_missing_input_behavior` is available when `pricesystem/price_select_rule/priceselect` is `4`.
- `pricesystem/price_select_rule/formula_fallback_strategy` is available when `pricesystem/price_select_rule/priceselect` is `4`.

Database Reference

Pricesystem coordinates the installed price providers. Most prices stay in the price-type modules (Customer Price, Price List, Category Price). The tables below are what the Pricesystem packages themselves add or extend.

Pricesystem core

Table / field	What it stores
`customer_group.pricesystem_price`	The group-level discount or price value entered on the customer-group form (Group Price).
`customer_group.pricesystem_price_type`	The application type chosen for the group value.
`pricesystem_price_adjustment_job`	Scheduled price-adjustment jobs (status, scope and target of a pending recalculation).
`quote_item.ps_discount_amount`, `quote_item.ps_applied_pricetype`	The resolved discount amount and the price type that won, stored on the cart line.
`sales_order_item.ps_discount_amount`, `sales_order_item.ps_applied_pricetype`	The same values carried onto the ordered line for reporting.
Customer attributes `pricesystem_price`, `pricesystem_price_type`	The per-customer value entered in the customer form's Pricesystem section.

Pricesystem AdvancedConfig add-on

Table / field	What it stores
`pricesystem_customer_sort_position`	Per-customer ordering of price types used by the configured selection strategy.
`pricesystem_customer_group_sort_position`	The corresponding ordering for customer groups.

Table / field	What it stores
`customer_group.priceselect`, `customer_group.price_formula`	Per-group price-selection rule and formula.
Customer attributes `verified`, `priceselect`, `price_formula`, `sortPrice…`	Per-customer selection rules, formulas and explicit sort positions.
Product attribute `discountable`	Whether a product participates in Pricesystem discounting.

Use the price-type modules' reference pages when planning pricing imports or integrations for the actual price records.

Common Issues

Start with one customer, one SKU and one quantity. Large grids and storefront pages add noise before the calculation itself is understood.

A custom price is not used

Check, in this order:

1. Price System is active for the product's website.
2. The price record is assigned to the current customer, group or matching rule.
3. The website and date range apply now.
4. The requested quantity reaches the intended break.
5. The price application type has a valid base price when it is relative.
6. The global or customer-specific selection strategy allows that price type to win.
7. The Price System cache has been cleaned after the change.

Inspect one calculation

List the available price types:

```
php bin/magento pricesystem:get-final-price --list-price-types
```

Calculate one SKU for one customer:

```
php bin/magento pricesystem:get-final-price \
  --customer-id=123 \
  --product-sku=24-MB01 \
  --qty=10
```

Use IDs and SKUs from your own test environment. The command can also accept a date and an explicit price type.

Capture a debug log

Debug logging is off by default. Turn it on only while you investigate a specific price calculation:

1. In the Magento Admin, open **Stores > Configuration > MageB2B > Pricesystem > Debug**.
2. Set **Enable log** to **Yes** and save the configuration. The underlying configuration path is `pricesystem/debug/enable_log`.
3. Reproduce the problem with one customer, one SKU and one quantity.
4. Review `var/log/pricesystem.log` for the price types considered, skipped or

selected during that request.

5. Set **Enable log** back to **No** when you have captured the calculation.

The log can contain a customer ID or `Guest`, the product SKU and the values used in pricing decisions. Treat it as store data when you share it with support, and include only the entries needed for the calculation you are investigating.

Warm the cache

Use a dry run before a broad warm-up:

```
php bin/magento pricesystem:cache:warmup \
  --customer-id=123 \
  --product-sku=24-MB01 \
  --dry-run
```

The command also supports customer groups, websites, dates, chunk sizes and an all-records mode. Do not start with `--all` on a large catalog.

Storefront prices do not refresh

- Check the browser network response for the Price System AJAX request.
- Confirm the theme keeps Magento price-box identifiers intact.
- Review configured skipped routes and chunk size.
- For Hyvä, confirm the Price System Hyvä package is installed and enabled.
- Clean both Magento full-page cache and `pricesystem_cache`.

Useful evidence for support

Include the SKU, customer ID or group, website, quantity, calculation date, expected price type and the output of the diagnostic command. Do not send customer passwords, access tokens or an unrestricted database export.

FAQ

Do I need to buy Price System separately?

No. Price System is the shared calculation layer installed with Customer Price, Price Lists, Category Price and Product-Customer Matrix. You buy the pricing product that fits the way you maintain prices; its installation includes the required Price System package.

What is a price type?

A price type is one source of a possible customer price, such as Customer Price, Pricelist, Base Pricelist, Category Price or Product-Customer Matrix. Each installed pricing product first resolves its own valid records. Price System then compares the resulting candidates and chooses the final value.

How does Price System choose the final price?

It collects candidates valid for the current product, customer, quantity, website and date. The configured strategy then chooses the lowest value, highest value, first valid type in a priority order, or a formula result when Advanced Config is installed.

This is the second selection stage. A price product may already have resolved several internal rows before it presents one candidate to Price System.

What is a base price list?

A base price list is a Price List marked **Is Base Pricelist**. It appears in Price System as the separate price type **Base Pricelist**. It is not attached to a particular dealer list and does not automatically fill gaps by itself.

For the usual fallback behavior, use **Price Priority Order** and place **Pricelist** before **Base Pricelist**. Price System then uses a valid regular-list price when one exists and continues to the base list when the regular-list type has no applicable price.

For example, a dealer list might contain a negotiated price for Product A, while the base list contains Product A and Product B. With that priority order, the dealer gets the negotiated value for Product A and the base-list value for Product B.

With **Lowest Price** or **Highest Price**, the two types are ordinary competing candidates. The base-list value can win even when a regular list also has a value. Keep base-list validation enabled so each website has one unambiguous list for this role.

Is Magento's catalog price still used?

Magento's catalog price remains the product's normal price and can be the reference for relative calculations and display. Whether a custom candidate replaces it depends on the enabled price types, calculation settings and final selection strategy.

How do quantity tiers work?

Stored price records can become valid from a specified quantity. Price System evaluates the current item quantity and uses the applicable break from each price type before comparing the resulting candidates. Test every threshold, including the quantity immediately below it.

What is the base price for a discount or surcharge?

Relative application types need a reference value from their owning pricing product and configuration. The global **Custom original price** also affects the reference used by shared calculations and display. A fixed application type already represents the final unit price and does not need a percentage base.

Can one customer use a different selection strategy?

Yes, with Advanced Config. It can override the global strategy for a customer or customer group and provide a custom type order or formula. A customer-specific override and a group override should be tested together so their precedence is intentional.

Why does My Prices differ from the product page?

The page can be limited to selected price types and can merge Base Pricelist rows into the Pricelist section. The storefront final price may also reflect cross-type selection, quantity and current product context. Treat My Prices as an account view of available stored prices, not as a substitute for testing checkout.

When should I clear or warm the Price System cache?

Normal Admin saves invalidate the relevant pricing data. Use targeted cache cleaning after imports, deployment or investigation of stale results. Cache warm-up is useful after large pricing changes when the first customer requests should not perform all calculations on demand.

What does Price Adjustment change?

Price Adjustment updates stored records in selected price types. Preview shows the affected rows and proposed values; applying the job writes those changes. It does not merely add a temporary storefront discount. Keep the queue consumer running and review the preview before submitting a large job.

Are calculated prices available through APIs?

Optional REST/SOAP and GraphQL add-ons expose pricing functions for integrations. The base package focuses on Magento calculation and storefront behavior. See Optional packages.

Why does the custom price column show "N/A" in the admin?

With Advanced Config, the column falls back to `N/A` when the reference price cannot be calculated — typically because the type configured as **Custom original price** is also restricted to discountable products and the product is not discountable. Magento warns about this combination when the Pricesystem configuration section is saved. See Advanced Config.

How do I investigate one unexpected price?

Use one customer, SKU, quantity, website and date. List every matching record by price type, then follow Common Issues. Enable the Price System log only for the short reproduction and disable it afterward.