Pricing System Overview

PDF Manual

MODULE

pricesystem

LAST UPDATED

2026-03-12

Table of Contents

Pricing System Overview · pricesystem

- Overview
- Pricing System Overview
- Getting Started
  - Installation
  - Configuration
- Features
  - Priority System
  - Quantity-Based Pricing
- Addons
  - Advanced Config
  - CSV Import/Export
  - SOAP / REST API
  - Sample Data
- Modules
  - Category Prices
  - Customer Prices
  - Price Lists
  - Product-Customer Matrix
- My Prices (Customer Account)
- Price Adjustment
  - How It Works
  - Run The Queue Consumer
  - Troubleshooting
- Price Selection
  - Customer Group Overrides
  - Customer Overrides
  - Formula Pricing
  - Sort Order Strategy
  - System Configuration (Global)
- Troubleshooting

Pricing System Overview

Pricing System Overview · /pricesystem

Pricesystem is MageB2B's pricing engine for Magento 2.

It consists of:

- the **core** package (required dependency for all price type modules), and
- optional **price type modules** (customerprice, categoryprice, pricelist, product customer matrix), distributed as separate Composer packages.

Packages And Modules

Core (Always Included)

- Composer package: `mageb2b/pricesystem`
- Magento module: `MageB2B_PricesystemCore`

Core provides:

- price selection strategies (Lowest / Highest / Sort order; Formula via add-on)
- the "My Prices" account page (price type modules can add CSV downloads)
- Price Adjustment (batch update jobs + queue consumer)
- price retrieval WebAPI endpoints in core (for example `/V1/pricesystem/get-final-price`); CRUD endpoints are provided by the API add-ons

Price Type Modules (Optional)

Price type	Composer package	Magento module	Price code
Customerprice	`mageb2b/pricesystem-customerprice`	`MageB2B_PricesystemCustomerprice`	`customer_price`
Categoryprice	`mageb2b/pricesystem-categoryprice`	`MageB2B_PricesystemCategoryprice`	`categoryprice`
Pricelist	`mageb2b/pricesystem-pricelist`	`MageB2B_PricesystemPricelist`	`pricelist` / `base_pricelist`
Product Customer Matrix	`mageb2b/pricesystem-productcustomermatrix`	`MageB2B_PricesystemProductCustomerMatrix`	`product_customer_matrix`

Key Topics

- Installation
- Configuration
- Price Selection
- My Prices
- Price Adjustment

Add-Ons

Add-On	Description
CSV Import/Export Add-On	Bulk management via CSV (per price type module)
SOAP / REST API Add-On	CRUD WebAPI endpoints (per price type module)
Advanced Config Add-On	Overrides, custom sort order, formula strategy

Add-On	Description
Sample Data	Demo entities and example prices

Price Type Guides

These pages explain how each price type behaves inside Pricesystem:

- Customerprice
- Categoryprice
- Pricelist
- Product Customer Matrix

Installation

Pricing System Overview · /pricesystem/getting-started/installation

Install Core (Always Required)

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem:*
php bin/magento module:enable MageB2B_PricesystemCore
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Install Price Type Modules (Optional)

Install and enable only the modules you bought/need.

Customerprice

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-customerprice:*
php bin/magento module:enable MageB2B_PricesystemCustomerprice
```

Categoryprice

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-categoryprice:*
php bin/magento module:enable MageB2B_PricesystemCategoryprice
```

Pricelist

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-pricelist:*
php bin/magento module:enable MageB2B_PricesystemPricelist
```

Product Customer Matrix

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
```

```
composer require mageb2b/pricesystem-productcustomermatrix:*
php bin/magento module:enable MageB2B_PricesystemProductCustomerMatrix
```

After enabling additional modules:

```
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Install Add-Ons (Optional)

Advanced Config Add-On

Adds customer/group override strategies, custom sort order, formula strategy, and verified/discountable checks.

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-advancedconfig:*
php bin/magento module:enable MageB2B_PricesystemCoreAdvancedConfig
php bin/magento setup:upgrade
php bin/magento cache:flush
```

CSV Import/Export Add-Ons (Per Price Type)

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-customerprice-importexport:*
composer require mageb2b/pricesystem-categoryprice-importexport:*
composer require mageb2b/pricesystem-pricelist-importexport:*
composer require mageb2b/pricesystem-productcustomermatrix-importexport:*
```

Enable the relevant modules and run upgrade:

```
php bin/magento module:enable MageB2B_PricesystemCustomerpriceImportExport
php bin/magento module:enable MageB2B_PricesystemCategorypriceImportExport
php bin/magento module:enable MageB2B_PricesystemPricelistImportExport
php bin/magento module:enable
MageB2B_PricesystemProductCustomerMatrixImpExp
php bin/magento setup:upgrade
php bin/magento cache:flush
```

SOAP / REST API Add-Ons (Per Price Type)

Prerequisite: install and enable the matching price type module first (Customerprice, Categoryprice, Pricelist, or Product Customer Matrix) before installing its API add-on.

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-customerprice-api:*
composer require mageb2b/pricesystem-categoryprice-api:*
composer require mageb2b/pricesystem-pricelist-api:*
composer require mageb2b/pricesystem-productcustomermatrix-api:*
```

Enable the relevant modules and run upgrade:

```
php bin/magento module:enable MageB2B_PricesystemCustomerpriceApi
php bin/magento module:enable MageB2B_PricesystemCategorypriceApi
php bin/magento module:enable MageB2B_PricesystemPricelistApi
php bin/magento module:enable MageB2B_PricesystemProductCustomerMatrixApi
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Verification

```
php bin/magento module:status | rg "MageB2B_Pricesystem"
```

Configuration

Pricing System Overview · /pricesystem/getting-started/configuration

Configure Pricesystem here:

- Admin -> Stores -> Configuration -> MageB2B -> Pricesystem

This page documents the core configuration keys. Some settings only appear when additional price type modules or add-ons are installed.

General Settings

Activate Pricesystem (Per Website)

- Config path: `pricesystem/general/active`
- Scope: Website

Hide Price Filter (Layered Navigation)

- Config path: `pricesystem/general/hide_price_filter`

Reload Prices With Ajax For Guests

- Config path: `pricesystem/general/reload_price_for_guest`
- Use this if you have Pricesystem prices configured for NOT_LOGGED_IN.

Routes To Skip Ajax Price Reload

- Config path: `pricesystem/general/skip_ajax_price_routes`
- Comma-separated list of routes (example: `cms_index_index`).

Use Native JSON Encode

- Config path: `pricesystem/general/use_native_json_encode`
- Enable if AJAX price requests return corrupted or empty JSON.

Price Cache Lifetime (Minutes)

- Config path: `pricesystem/general/price_cache_lifetime`

Price Calculation

Custom Original Price

- Config path: `pricesystem/price_calculation/custom_original_price`
- Select which price type is treated as "original" price for downstream calculations and strike-through behavior.

Skip Greater Tier Prices

- Config path: `pricesystem/price_calculation/skip_greater_tier_price`

- If enabled, tier prices are ignored when they are greater than or equal to the current final price.

Disable Pricesystem For Selected Prices

- Config path: `pricesystem/price_calculation/disabled_prices`
- If a price type is disabled here, Magento default pricing is used for that price type on the frontend.

Skip Zero Price

- Config path: `pricesystem/price_calculation/skip_zero_price`
- If enabled, prices `<= 0` are ignored by selection and by formula inputs.

Additional Discounts (Customer / Group)

Apply customer/group discounts on selected price types:

- Customer discount on: `pricesystem/price_calculation/apply_customer_discount_on`
- Group discount on: `pricesystem/price_calculation/apply_group_discount_on`
- Which discount wins first: `pricesystem/price_calculation/first_priority_customer_discount_or_group`

Display Settings

Show "My Prices" Under My Account

- Config path: `pricesystem/display/custom_prices_my_account`

Related options:

- Dropdown restriction: `pricesystem/display/custom_prices_my_account_provider_types`
- Show only when prices exist: `pricesystem/display/custom_prices_my_account_not_empty`

Details: My Prices

Standard Price Strike-Through

- Config path: `pricesystem/display/standard_price_strikethrough`

Tier Prices

- Show tier prices: `pricesystem/display/show_tierprices`
- Show tier prices on list page: `pricesystem/display/show_tierprices_on_listpage`

Discount Text

- Product view: `pricesystem/display/frontend_show_discount_percent_product_view`
- List page: `pricesystem/display/frontend_show_discount_percent_product_list`
- Discount text template: `pricesystem/display/frontend_discount_text`
- Configurable list page text:

`pricesystem/display/frontend_conf_product_discount_text_listpage`
- Optional CSS style: `pricesystem/display/frontend_discount_text_style`

Hide Prices For Customer Groups (List Page)

- Config path: `pricesystem/display/hide_price_on_list_page_customer_group`

Price Selection Rules

Global strategy:

- Config path: `pricesystem/price_select_rule/priceselect`
- Lowest (1), Highest (2), Sort order (3)
- Formula (4) requires Advanced Config Add-On

Sort order list:

- Config path: `pricesystem/price_select_rule/price_sort_order`

Strike-through behavior for sort order:

- Next available strike-through: `pricesystem/price_select_rule/use_next_available_price_in_strike_through`
- Skip discounts on strike-through: `pricesystem/price_select_rule/skip_customer_group_discounts_for_next_available_price`

Details: Price Selection

API Settings (Sync Fields)

If you synchronize Pricesystem data via external systems, you can configure which API fields are used:

- Use custom API fields: `pricesystem/api/use_custom_api_fields`
- Customer attribute: `pricesystem/api/customer_attribute`
- Customer group attribute: `pricesystem/api/customer_group_attribute`
- Product attribute: `pricesystem/api/product_attribute`
- Category attribute: `pricesystem/api/category_attribute`

Debug Log

- Enable: `pricesystem/debug/enable_log`
- Log file: `var/log/pricesystem.log`

Cache Type

Pricesystem registers a dedicated cache type:

- Cache type code: `pricesystem_cache`

Example commands:

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

```
php bin/magento cache:flush
```

Module-Specific Settings

Some settings appear only when the corresponding price type module is installed:

- Customerprice settings: `pricesystem/customerprice/*`
- Categoryprice settings: `pricesystem/categoryprice/*`
- Pricelist settings: `pricesystem/pricelist/*`
- Product Customer Matrix settings: `pricesystem/productcustomermatrix/*`

See the dedicated KB pages for details:

- `/kb/pricesystem/modules/customer-prices`
- `/kb/pricesystem/modules/category-prices`
- `/kb/pricesystem/modules/pricelists`
- `/kb/pricesystem/modules/product-customer-matrix`

Priority System

Pricing System Overview · /pricesystem/features/priority-system

The Pricing System uses configurable selection strategies to determine which price applies when multiple rules match.

How It Works

When a customer views a product, the system:

1. Collects all matching prices from:
    - Customer Prices
    - Category Prices
    - Price Lists
    - Product-Customer Matrix
2. Applies validity checks (dates, website, quantity)
3. Resolves conflicts using configured strategy
4. Returns the winning price

Priority Levels

Priority fields are available in Category Prices, Price Lists, and Product-Customer Matrix (not Customer Prices). The ranges below are team conventions, not hard-enforced system limits:

Priority Range	Typical Use
0-10	Base/default pricing
11-30	Group/segment pricing
31-50	Promotional pricing
51-70	Contract pricing
71-99	VIP/override pricing

> **Note**: Higher numbers = higher priority where a priority field exists.

Selection Strategies

1. Lowest Price (Default)

The system selects the lowest resulting price value.

Example:

- Customer Price: $100
- Pricelist Price: $90
- **Winner**: Pricelist Price ($90)

2. Highest Price

The system selects the highest resulting price value.

Example:

- Customer Price: $100
- Pricelist Price: $90
- **Winner**: Customer Price ($100)

3. Sort Order

Define your own price type ordering in admin:

1. Navigate to **Stores > Config > Pricesystem**
2. Find **Price Priority Order**
3. Drag-and-drop price types to set order

4. Formula (Advanced Config Add-On)

Uses formula-based calculation logic instead of direct lowest/highest/sort-order selection.

Module-Specific Strategies

Category Prices

Extra selection strategies for customer vs group conflicts:

Strategy	Behavior
Select price by priority (`CHOOSE_BY_PRIORITY`)	Compare priorities
GROUP_PRICE_FIRST	Always use group price
CUSTOMER_PRICE_FIRST	Always use customer price

Price Lists

Merge behavior for list conflicts:

- **Disabled**: Lower-priority lists are ignored; same highest-priority lists can still combine qty tiers
- **Enabled**: Lower-priority lists can also participate in qty-tier merge

Configuration

Core Priority Settings

Stores > Configuration > MageB2B > Pricesystem > Price Calculation:

Setting	Description
First Priority	Customer vs Group discount order

Stores > Configuration > MageB2B > Pricesystem > Price Select Rule settings:

Setting	Description
Price Select Rule	Lowest / Highest / Sort order / Formula (Advanced Config Add-On)

Category Price Settings

Stores > Configuration > MageB2B > Pricesystem > Categoryprice settings:

Setting	Description
Price Select Rule	Select price by priority / Group First / Customer First

Pricelist Settings

Stores > Configuration > MageB2B > Pricesystem > Pricelist settings:

Setting	Description
Merge Pricelist Qtys	Controls whether lower-priority lists can join qty-tier merge

Best Practices

1. **Document your priority scheme** in team wiki
2. **Use consistent ranges** across all modules
3. **Test edge cases** with overlapping rules
4. **Audit regularly** for conflicting priorities
5. **Treat ranges as team convention**; system limits depend on module schema

Troubleshooting

Wrong Price Displayed

1. Check priority values of all matching rules
2. Verify date validity of all rules
3. Check website scope of rules
4. Review selection strategy configuration

Enable Debug Logging

Stores > Config > Pricesystem > Debug > Enable log: Yes

Check `var/log/pricesystem.log` for price calculation details.

Quantity-Based Pricing

Pricing System Overview · /pricesystem/features/quantity-tiers

All pricing modules support quantity-based tiers, enabling volume discounts across customer prices, category prices, pricelists, and matrix rules.

How It Works

Each price entry includes a `qty` field representing the minimum quantity threshold:

Qty	Price	Applied When
1	$100	Cart qty 1-9
10	$95	Cart qty 10-49
50	$90	Cart qty 50-99
100	$80	Cart qty 100+

The system selects the tier where cart quantity >= tier qty.

Module Support

Customer Prices

Create multiple entries for same customer/product with different qty values.

Category Prices

Apply qty tiers to entire categories for customers or groups.

Price Lists

Each product in a pricelist can have multiple qty entries.

> **Note**: "Multi-Qty Price" is checked in the pricelist API save flow for qty-handling behavior.

Product-Customer Matrix

Qty field in main matrix applies to all matching products.

Configuration

Skip Greater Tier Prices

Stores > Config > Pricesystem > Price Calculation > Skip Greater Tier Prices

When enabled: If qty tier price is greater than or equal to the current final price, the tier is skipped.

Prevents accidental price increases on bulk orders.

Price List Merge

Stores > Config > Pricesystem > Pricelist Settings > Merge Pricelist Qtys

When disabled, lower-priority pricelists are ignored (same highest-priority lists can still combine qty tiers). When enabled, lower-priority pricelists can also join the qty-tier merge.

Creating Qty Tiers

Admin Example (Customer Prices)

For Widget-X and Customer ABC:

1. Add price: Product = Widget-X, Customer = ABC, Qty = 1, Price = $100
2. Add price: Product = Widget-X, Customer = ABC, Qty = 10, Price = $95
3. Add price: Product = Widget-X, Customer = ABC, Qty = 50, Price = $90

CSV Import Example

```
product_sku,customer_email,qty,price,website_id
WIDGET-X,abc@example.com,1,100.00,0
WIDGET-X,abc@example.com,10,95.00,0
WIDGET-X,abc@example.com,50,90.00,0
```

Frontend Display

Tier prices display on product pages showing all available breaks:

> Buy 1-9: $100 each Buy 10-49: $95 each (5% savings) Buy 50+: $90 each (10% savings)

Configuring Display

Stores > Configuration > MageB2B > Pricesystem > Display settings:

- Show tierprices: Yes/No
- Display discount text on product view pages: Yes/No
- Display discount text on listing pages: Yes/No

Best Practices

1. **Plan tiers strategically**: Common breaks are 1, 10, 25, 50, 100, 500
2. **Ensure descending prices**: Each tier should be lower than previous
3. **Document in contracts**: Match tiers to customer agreements
4. **Test edge quantities**: Verify behavior at tier boundaries

Troubleshooting

Tier Not Applying

- Check qty value matches or exceeds tier threshold
- Verify date validity if applicable
- Check website scope
- Review priority if multiple rules exist

Wrong Tier Selected

- System uses highest qty tier that cart qty meets
- E.g., Cart qty 25 with tiers at 1, 10, 50 uses qty 10 tier

Advanced Config

Pricing System Overview · /pricesystem/addons/advanced-config

The Advanced Config add-on extends Pricesystem with additional configuration and override options, mainly around **price selection**.

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-advancedconfig:*

php bin/magento setup:upgrade
php bin/magento cache:flush
```

Magento module: `MageB2B_PricesystemCoreAdvancedConfig`

What It Adds

- **Formula strategy** in price selection (for configurable, calculated pricing)
- **Customer overrides** for how Pricesystem selects prices
- **Customer group overrides** for how Pricesystem selects prices
- **Custom sort order** per customer / customer group (when using the Sort order strategy)
- Additional checks that affect applicability of prices (for example "verified" / "discountable" flags, depending on your setup)

Where To Configure

- Global defaults: Stores > Configuration > MageB2B > Pricesystem > Price select rule settings
- Overrides (customer group): Customers > Customer Groups > Edit
- Overrides (customer): Customers > All Customers > Edit

Related

- Price Selection
- Formula Pricing

CSV Import/Export

Pricing System Overview · /pricesystem/addons/csv-import-export

Bulk manage pricing data via CSV files (import + export).

Overview

The CSV Import/Export add-on provides:

- CLI commands to import/export data per price type
- Magento Import/Export entities (System > Data Transfer > Import)
- CSV downloads on the **My Prices** page are provided by the base price-type modules (see My Prices)

This is a per-module add-on: install it only for the price types you use.

Installation

Install the add-on for each module you need:

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-customerprice-importexport:*
composer require mageb2b/pricesystem-categoryprice-importexport:*
composer require mageb2b/pricesystem-pricelist-importexport:*
composer require mageb2b/pricesystem-productcustomermatrix-importexport:*

php bin/magento setup:upgrade
php bin/magento cache:flush
```

Import/Export In Admin (Magento ImportExport)

You can import via Magento's built-in importer:

1. Go to **System > Data Transfer > Import**
2. Choose one of the entity types below
3. Upload a CSV and run the import

Entity types (as they appear in Magento):

- Pricesystem Customerprice
- Pricesystem Categoryprice
- Pricesystem Categoryprice Customergroup
- Pricesystem Pricelist
- PricesystemProductCustomerMatrix

Tip: If you are unsure about the exact column names for your version, do an export first and use the exported CSV as your template.

Import/Export Via CLI

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

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

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

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

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

Useful CLI Options

Imports support common ImportExport flags (example):

```
php bin/magento pricesystem:import-customerprice /path/to/customerprice.csv
\
  --behavior=add_update \
  --field_separator=, \
  --fields_enclosure='"'
```

Exports support output path + CSV formatting (example):

```
php bin/magento pricesystem:export-customerprice \
  --export_path=var/export/customerprice.csv \
  --field_separator=, \
  --fields_enclosure='"'
```

Customer CSV Download (My Prices)

If enabled, customers can download their prices from **My Account > My Prices**.

CSV formatting for downloads is configured per price type:

- delimiter: `pricesystem/<type>/csv_delimeter` (note the spelling)
- enclosure: `pricesystem/<type>/csv_encloser`

Best Practices

1. **Validate before import**: Run Magento's import validation (System > Data Transfer > Import) before you import into production
2. **Backup existing data**: Export before bulk updates

3. **Use staging**: Test imports in staging environment
4. **Small batches**: For large imports, split into batches
5. **UTF-8 encoding**: Always save CSV as UTF-8

Related

- Customer Prices
- SOAP / REST API

SOAP / REST API

Pricing System Overview · /pricesystem/addons/rest-api

Adds WebAPI endpoints for integrations (REST + SOAP).

Overview

Pricesystem ships with core **read** endpoints for calculated prices (see "Core endpoints" below).

This add-on provides **CRUD** endpoints per price type module:

- Customerprice API (`mageb2b/pricesystem-customerprice-api`)
- Categoryprice API (`mageb2b/pricesystem-categoryprice-api`)
- Pricelist API (`mageb2b/pricesystem-pricelist-api`)
- Product Customer Matrix API (`mageb2b/pricesystem-productcustomermatrix-api`)

Installation

Install the add-on for each module:

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-customerprice-api:*
composer require mageb2b/pricesystem-categoryprice-api:*
composer require mageb2b/pricesystem-pricelist-api:*
composer require mageb2b/pricesystem-productcustomermatrix-api:*

php bin/magento setup:upgrade
php bin/magento cache:flush
```

Core API Endpoints

The core package `mageb2b/pricesystem` provides calculated price endpoints:

REST URL note:

- REST routes use the base path `/rest` and an optional store code:
  - `/rest/V1/...` (default)
  - `/rest/{storeCode}/V1/...` (store view scope)

Method	URL	Notes
POST	`/V1/pricesystem/get-final-price`	Admin context (ACL: `MageB2B_PricesystemCore::main`)
POST	`/V1/pricesystem/get-multi-price`	Admin context (ACL: `MageB2B_PricesystemCore::main`)
POST	`/V1/pricesystem/me/get-final-price`	Customer token only (`self`)
POST	`/V1/pricesystem/me/get-multi-price`	Customer token only (`self`)

Method	URL	Notes
POST	`/V1/pricesystem/context/get-final-price`	Context-based (`anonymous`)
POST	`/V1/pricesystem/context/get-multi-price`	Context-based (`anonymous`)

Authentication / Permissions

Pricesystem uses Magento's standard WebAPI authentication and ACL. The required permission depends on the endpoint:

- Core admin endpoints require the ACL resource `MageB2B_PricesystemCore::main`
- `/me/...` endpoints require a customer token (`self`)
- `/context/...` endpoints are available for `anonymous` (context-based)

Get An Admin Token (Example)

```
curl -X POST "https://your-store.com/rest/V1/integration/admin/token" \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"***"}'
```

Then use it:

```
curl -H "Authorization: Bearer YOUR_TOKEN" \
  "https://your-store.com/rest/V1/pricesystem/customerprice/search"
```

CRUD Endpoints (Add-On)

These endpoints are added by installing the API package for the corresponding price type module.

Customerprice (`mageb2b/pricesystem-customerprice-api`)

Method	URL
GET	`/V1/pricesystem/customerprice/:id`
GET	`/V1/pricesystem/customerprice/search`
POST	`/V1/pricesystem/customerprice`
PUT	`/V1/pricesystem/customerprice/:id`
POST	`/V1/pricesystem/customerprice/savebulk`
DELETE	`/V1/pricesystem/customerprice/:id`
DELETE	`/V1/pricesystem/customerprice/deleteByCustomerId/:customerId`
POST	`/V1/pricesystem/customerprice/deletebulk`

Categoryprice (`mageb2b/pricesystem-categoryprice-api`)

Method	URL
GET	`/V1/pricesystem/categoryprice/:id`

Method	URL
GET	`/V1/pricesystem/categoryprice/search`
POST	`/V1/pricesystem/categoryprice`
PUT	`/V1/pricesystem/categoryprice/:id`
DELETE	`/V1/pricesystem/categoryprice/:id`
GET	`/V1/pricesystem/categoryprice/customergroup/:id`
GET	`/V1/pricesystem/categoryprice/customergroup/search`
POST	`/V1/pricesystem/categoryprice/customergroup`
PUT	`/V1/pricesystem/categoryprice/customergroup/:id`
DELETE	`/V1/pricesystem/categoryprice/customergroup/:id`

Pricelist (`mageb2b/pricesystem-pricelist-api`)

Method	URL
GET	`/V1/pricesystem/pricelist/:id`
GET	`/V1/pricesystem/pricelist/search`
POST	`/V1/pricesystem/pricelist`
PUT	`/V1/pricesystem/pricelist/:id`
POST	`/V1/pricesystem/pricelistbulk`
DELETE	`/V1/pricesystem/pricelist/:id`
DELETE	`/V1/pricesystem/pricelist/all`
GET	`/V1/pricesystem/pricelistproduct/:id`
GET	`/V1/pricesystem/pricelistproduct/search`
POST	`/V1/pricesystem/pricelistproduct`
POST	`/V1/pricesystem/pricelistproductbulk`
PUT	`/V1/pricesystem/pricelistproduct/:id`
PUT	`/V1/pricesystem/pricelistproductbulk`
DELETE	`/V1/pricesystem/pricelistproduct/:id`
GET	`/V1/pricesystem/pricelistcustomer/:id`
GET	`/V1/pricesystem/pricelistcustomer/search`
POST	`/V1/pricesystem/pricelistcustomer`
PUT	`/V1/pricesystem/pricelistcustomer/:id`
DELETE	`/V1/pricesystem/pricelistcustomer/:id`
DELETE	`/V1/pricesystem/pricelistcustomer/deleteByCustomerId/:customerId`
GET	`/V1/pricesystem/pricelistgroup/:id`
GET	`/V1/pricesystem/pricelistgroup/search`
POST	`/V1/pricesystem/pricelistgroup`

Method	URL
PUT	`/V1/pricesystem/pricelistgroup/:id`
DELETE	`/V1/pricesystem/pricelistgroup/:id`

Product Customer Matrix (`mageb2b/pricesystem-productcustomermatrix-api`)

Method	URL
GET	`/V1/pricesystem/productcustomermatrix/:id`
GET	`/V1/pricesystem/productcustomermatrix/search`
POST	`/V1/pricesystem/productcustomermatrix`
PUT	`/V1/pricesystem/productcustomermatrix/:id`
DELETE	`/V1/pricesystem/productcustomermatrix/:id`
GET	`/V1/pricesystem/productcustomermatrixattribute/:id`
GET	`/V1/pricesystem/productcustomermatrixattribute/search`
POST	`/V1/pricesystem/productcustomermatrixattribute`
PUT	`/V1/pricesystem/productcustomermatrixattribute/:id`
DELETE	`/V1/pricesystem/productcustomermatrixattribute/:id`
GET	`/V1/pricesystem/productcustomermatrixcustomer/:id`
GET	`/V1/pricesystem/productcustomermatrixcustomer/search`
POST	`/V1/pricesystem/productcustomermatrixcustomer`
PUT	`/V1/pricesystem/productcustomermatrixcustomer/:id`
DELETE	`/V1/pricesystem/productcustomermatrixcustomer/:id`

Related

- CSV Import/Export

Sample Data

Pricing System Overview · /pricesystem/addons/sample-data

Sample data packages install demo entities (products/customers) and example pricing records so you can explore Pricesystem features quickly.

Packages

Core sample data (required base package):

- Composer package: `mageb2b/pricesystem-sample-data`
- Magento module: `MageB2B_PricesystemCoreSampleData`

Sample data for price types (optional on top of core sample data):

- `mageb2b/pricesystem-customerprice-sample-data` (`MageB2B_PricesystemCustomerpriceSampleData`)
- `mageb2b/pricesystem-categoryprice-sample-data` (`MageB2B_PricesystemCategorypriceSampleData`)
- `mageb2b/pricesystem-pricelist-sample-data` (`MageB2B_PricesystemPricelistSampleData`)
- `mageb2b/pricesystem-productcustomermatrix-sample-data` (`MageB2B_PricesystemProductCustomerMatrixSampleData`)

What Gets Installed (High Level)

Core sample data installs:

- categories
- customer groups
- customers (+ addresses)
- products

Each price type sample data installs example pricing rows for that module (for example: customer prices, category prices, pricelists, matrices).

Installation

```
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/
composer require mageb2b/pricesystem-sample-data:*

# Optional: add sample data for specific price types
composer require mageb2b/pricesystem-customerprice-sample-data:*
composer require mageb2b/pricesystem-categoryprice-sample-data:*
composer require mageb2b/pricesystem-pricelist-sample-data:*
composer require mageb2b/pricesystem-productcustomermatrix-sample-data:*

php bin/magento setup:upgrade
```

```
php bin/magento cache:flush
```

Note: Sample data patches are applied once. If you want to reinstall sample data on an existing environment, do it in a fresh database or remove the sample data rows first.

Category Prices

Pricing System Overview · /pricesystem/modules/category-prices

Set prices at the category level to affect hundreds of products with a single rule. Supports both individual customer and customer group pricing.

Overview

Category Prices provide:

- **Bulk efficiency**: One rule affects all products in a category
- **Dual support**: Customer-specific AND group-based pricing
- **Priority system**: Conflict resolution with numeric priorities
- **Selection strategies**: Three ways to resolve customer vs group conflicts

Two Pricing Tables

Customer Category Prices

Table: `pricesystem_categoryprice`

Assign category prices to individual customers.

Group Category Prices

Table: `pricesystem_categoryprice_customergroup`

Assign category prices to customer groups (Wholesale, Retail, VIP, etc.)

Creating Category Prices

For Individual Customers

1. Navigate to **Pricesystem > Category Prices > Customer**
2. Click **Add New**
3. Select:
    - **Category** from searchable selector
    - **Customer** (searchable)
    - **Quantity** threshold
    - **Price** value
    - **Price Application Type**
    - **Base Price** (optional)
    - **Website** scope
    - **Date range** (optional)
4. Save

For Customer Groups

1. Navigate to **Pricesystem > Category Prices > Customer Groups**
2. Click **Add New**

3. Select:
    - **Category** from searchable selector
    - **Customer Group** (Wholesale, Retail, etc.)
    - **Quantity** threshold
    - **Price** value
    - **Price Application Type**
    - **Base Price** (optional)
    - **Website** scope
    - **Date range** (optional)
4. Save

Priority Notes

Priority exists in the data model, but in this module version the standard admin forms do not expose a priority input field.

When both customer and group category prices are present for the same context, selection follows the configured category-price select rule.

Equal-result tie-break behavior can still depend on global Pricesystem selection strategy settings.

Selection Strategies

Configure at **Pricesystem > Categoryprice Settings > Price Select Rule**:

1. Select price by priority (Default)

Compare priority fields; higher wins.

2. GROUP_PRICE_FIRST

Always use group price, ignore customer price.

3. CUSTOMER_PRICE_FIRST

Always use customer price, ignore group price.

Use Case Examples

Wholesale Category Discount

All products in "Bulk Materials" category get 40% off for wholesale group:

- Category: Bulk Materials
- Group: Wholesale
- Price Application: Percentage (40%)
- Priority: 10 (for integration/import-driven setups where priority is managed outside the standard form)

VIP Override

Customer ABC gets special pricing on electronics:

- Category: Electronics
- Customer: ABC Corp
- Price: Fixed ($500 per product in matching category)
- Priority: 30 (for integration/import-driven setups where priority is managed outside the standard form)

Best Practices

1. **Use groups for bulk** - Easier maintenance
2. **Reserve high priority** for special customers if you manage priority through integration/import
3. **Document priority scheme** in internal wiki
4. **Test with staging** before production changes

Related

- Priority System
- Customer Prices

Customer Prices

Pricing System Overview · /pricesystem/modules/customer-prices

Assign unique product prices to individual customers with full support for quantity tiers, date ranges, and multi-website scoping.

Overview

Customer Prices allow you to:

- Set specific prices per customer per product
- Define quantity-based price breaks
- Schedule prices with start/end dates
- Scope prices to specific websites

Database Structure

Column	Type	Description
entity_id	INT	Product ID
customer_id	INT	Customer ID
qty	DECIMAL	Quantity threshold
value	DECIMAL	Price value
from_date	DATE	Valid from
to_date	DATE	Valid until
price_application_type	TINYINT	Price application mode (fixed/surcharge/discount/%).
price_attribute_id	SMALLINT	Optional base price attribute used for calculations.
website_id	INT	Website scope

Creating Customer Prices

Via Admin Panel

1. Navigate to **Pricesystem > Customerprice > All Customerprices**
2. Click **Add New**
3. Fill in the form:
    - Select **Product** (searchable dropdown)
    - Select **Customer** (searchable dropdown)
    - Enter **Quantity** (minimum qty for this price)
    - Enter **Price** value
    - Select **Price Application Type** (Fixed / Surcharge / Surcharge % / Discount / Discount %)
    - Select **Base Price** (optional)
    - Set **From Date** and **To Date** (optional)
    - Select **Website** scope
4. Click **Save Customerprice**

Quantity Tiers Example

To set up volume discounts for Customer ABC on Product XYZ:

Qty	Price	Notes
1	$100	Standard price
10	$95	5% discount
50	$90	10% discount
100	$80	20% discount

Create 4 entries with the same customer/product but different qty values.

Time-Based Pricing

For promotional periods, set date ranges:

- **Contract pricing**: from_date = contract start, to_date = contract end
- **Seasonal pricing**: from_date = season start, to_date = season end
- **No dates**: Price always active

Customer "My Prices" Feature

If My Prices is enabled and the customer is logged in, they can view and download assigned prices:

1. Customer logs into their account
2. Navigates to **My Prices** section
3. Selects a price type and views available prices for that selection
4. Downloads prices as CSV (if enabled)

Enable at: **Stores > Configuration > MageB2B > Pricesystem > Customerprice settings > Enable download link**

Validation Rules

- **Frontend scope**: Guest sessions do not resolve customer-specific prices; API/admin flows can resolve a customer explicitly by ID
- **Unique constraint**: (entity_id, customer_id, qty, from_date, to_date, website_id)
- **Date validation**: date format is validated; there is no enforced `from_date <= to_date` rule in the current save flow

Use Cases

Contract Pricing

Set negotiated prices for specific customers with contract validity dates.

Volume Discounts

Reward bulk purchases with quantity-based tier pricing.

Seasonal Promotions

Time-limited pricing for specific customers during promotional periods.

Regional Pricing

Different prices per website for multi-region operations.

Best Practices

1. **Use date ranges** for contract pricing to auto-expire
2. **Set website scope** for regional pricing
3. **Document qty tiers** in customer contracts
4. **Regular audits** to clean expired prices

Related

- Quantity Tiers Feature
- Priority System
- CSV Import/Export Add-On

Price Lists

Pricing System Overview · /pricesystem/modules/pricelists

Create and manage named price lists with automatic customer assignment, validity periods, and priority-based selection.

Overview

Price Lists provide:

- **Named entities**: Give meaningful names to pricing agreements
- **Auto-assignment**: Assign customers based on attributes
- **Base pricelist**: Separate `base_pricelist` price type
- **Date validity**: Both pricelist and product-level dates
- **Multi-level assignment**: Customer, group, or attribute-based

Database Structure

Main Tables

Table	Purpose
`pricesystem_pricelist`	Pricelist definitions
`pricesystem_pricelist_product`	Product prices within lists
`pricesystem_pricelist_customer`	Direct customer assignments
`pricesystem_pricelist_group`	Group assignments

Pricelist Fields

Field	Description
name	Unique pricelist name
active	Enable/disable
priority	Conflict resolution
from_date / to_date	Validity period
is_base_pricelist	Mark as `base_pricelist` type
customer_attribute	Auto-assignment attribute
customer_attribute_value	Value to match

Creating a Pricelist

1. Navigate to **Pricesystem > Pricelist > All Pricelist**
2. Click **Add New**
3. **General Information**:
    - Name: "Enterprise Q1 2025"
    - Active: Yes

    - Priority: 20
    - From Date: 2025-01-01
    - To Date: 2025-03-31
    - Is Base Pricelist: No
4. **Products Tab**:
    - Add products with qty and price
    - Set product-specific validity dates
5. **Customers Tab**:
    - Assign specific customers
6. **Groups Tab**:
    - Assign customer groups
7. Save

Assignment Methods

1. Direct Customer Assignment

Manually assign customers to pricelists via admin.

2. Group-Based Assignment

All customers in a group inherit the pricelist.

3. Attribute-Based Auto-Assignment

At registration, customers are auto-assigned based on:

- Customer attribute (e.g., company_type = "enterprise")
- Address attribute (e.g., country = "US")
- Can require both (AND) or either (OR)

Enable at: **Pricesystem > Pricelist Settings > Enable Auto-Assignment**

Note: In this code snapshot, the registration observer reads a legacy config key for this toggle. If auto-assignment does not trigger, verify the installed module version and key mapping.

Base Pricelist Feature

Mark one pricelist as "Base" to serve as default:

- Base pricelist is treated as a separate price type and still requires customer/group assignment
- Uniqueness per website is enforced when **Enable base pricelist validation** is enabled

Merge Behavior

When multiple pricelists have the same priority:

- **Disabled**: Lower-priority pricelists are ignored; highest-priority matches are still combined per qty
- **Enabled**: All matched pricelists are considered; per-qty value uses priority and then global strategy for ties

Configure at: **Pricesystem > Pricelist Settings > Merge Pricelist Qtys**

Use Cases

Contract Pricing

- Name: "Acme Corp 2025 Contract"
- Dates: Contract period
- Products: Contracted items with agreed prices
- Customer: Acme Corp only

Seasonal Campaign

- Name: "Summer Sale 2025"
- Dates: June 1 - August 31
- Products: Summer items with discounts
- Groups: All retail customers

Volume Tiers

- Name: "Distributor Standard"
- Products: Each with qty tiers (1-99, 100-499, 500+)
- Groups: Distributor group
- Is Base: Yes

Related

- Priority System
- Quantity Tiers

Product-Customer Matrix

Pricing System Overview · /pricesystem/modules/product-customer-matrix

The most flexible pricing component for complex contract pricing using a three-layer architecture with AND/OR attribute logic.

Overview

The Matrix provides:

- **Attribute-based rules**: Match products by configured filterable attributes
- **AND/OR logic**: Combine conditions flexibly
- **Customer segments**: Group customers by attributes
- **Customer-only pricing**: Applied for logged-in/assigned customers
- **Priority resolution**: Clear conflict handling
- **Date validity**: Time-limited pricing rules

Three-Layer Architecture

Layer 1: Main Matrix

Core pricing rule with:

- Name (unique identifier)
- Price and quantity
- Priority level
- Validity dates
- Customer attribute for segmentation

Layer 2: Product Attributes

Define which products match:

- Attribute code (sku, category_ids, color, etc.)
- Attribute value to match
- AND/OR relation between conditions

Layer 3: Customer Assignments

Assign to specific customers:

- Direct customer ID assignment
- Customer-specific validity dates
- Overrides segment-based matching

Creating a Matrix Price

Via Admin Panel

1. Navigate to **Pricesystem > Product Customer Matrix**

2. Click **New Matrix**
3. **General Tab**:
    - Name: "Enterprise Electronics Deal"
    - Price: $500
    - Price Application Type: Defines how `price` is applied (e.g., fixed value, discount %)
    - Quantity: 1
    - Priority: 30
    - Valid From/To: Contract dates
4. **Product Attributes Tab**:
    - Attributes Relation: AND
    - Add: category_ids = "23" (stored category value/ID)
    - Add: brand = "premium"
5. **Customers Tab**:
    - Add specific customers
    - Set customer-specific dates
6. Save

AND/OR Logic

AND Mode (All Must Match)

All attribute conditions must be true:

```
category_ids = "23" AND brand = "premium"
```

Only products matching both stored values match.

OR Mode (Any Can Match)

Any attribute condition can be true:

```
category_ids = "23" OR category_ids = "24"
```

Products matching either stored category value match.

Customer Segmentation

Instead of assigning individual customers, use customer attributes:

1. Set `customer_attribute_value` in main matrix
2. E.g., "enterprise" for all enterprise customers
3. Any customer with matching attribute gets the price

Priority Resolution

When multiple matrices match:

1. Direct customer assignments and customer attribute matches can both apply
2. Priority influences which matched matrix overrides others
3. Keep overlapping rules minimal and priorities explicit to avoid ambiguity

Use Cases

Multi-Product Contract

"Customer ABC gets 20% off all Brand X products":

- Attributes: brand = "X" (OR mode for multiple brands)
- Customers: ABC Corp
- Price Application Type: Discount %
- Price: 20
- Priority: 40

Regional Segment Pricing

"All US distributors get special pricing on category Y":

- Customer Attribute: region = "US"
- Attributes: category_ids = "15" (stored category value/ID)
- Price: $450
- Priority: 20

SKU-Specific Override

"Product SKU-123 special price for Customer XYZ":

- Attributes: sku = "SKU-123"
- Customers: XYZ Corp
- Price: $99
- Priority: 50 (highest)

Best Practices

1. **Use segments** for broad rules, direct assignment for exceptions
2. **Set explicit priorities** and avoid unnecessary overlap between matrices
3. **Document matrix rules** in customer contracts
4. **Test combinations** in staging environment

Related

- Priority System
- CSV Import/Export

My Prices (Customer Account)

Pricing System Overview · /pricesystem/my-prices

Pricesystem Core adds a "My Prices" page in the customer account.

URL (frontend):

- `/pricesystem/index/index`

Route behavior:

- If not logged in: redirects to `customer/account/login`
- If logged in and page is unavailable: redirects to `customer/account`

Enable / Disable

- Config path: `pricesystem/display/custom_prices_my_account`
- Label in admin: "Show my prices under my account"
- Page availability is based on this setting, with additional module plugin overrides possible.

Restrict Dropdown Options

You can restrict which price types show up in the dropdown:

- Config path: `pricesystem/display/custom_prices_my_account_provider_types`

If empty, all installed price types can be shown.

Hide When No Prices Exist

- Config path: `pricesystem/display/custom_prices_my_account_not_empty`

If enabled, this hides the "My Prices" link in account navigation when no prices are assigned (this can be extended by price-type modules).

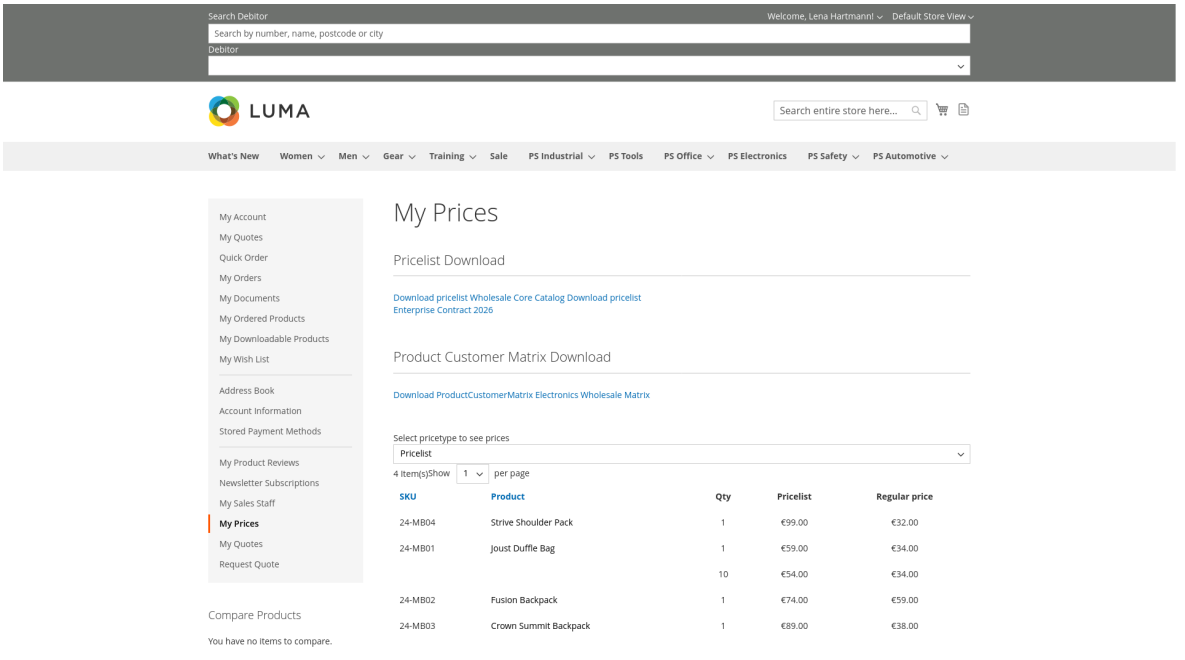
CSV Downloads (Per Price Type)

The core "My Prices" page can expose CSV downloads, but the actual download features come from the installed price type modules:

- Customerprice: `pricesystem/customerprice/enable_download`
- Categoryprice: `pricesystem/categoryprice/enable_download`
- Pricelist: `pricesystem/pricelist/enable_download`
- Product Customer Matrix: `pricesystem/productcustomermatrix/enable_download`

Each price type has its own CSV formatting settings:

- delimiter: `pricesystem/<type>/csv_delimeter` (note the spelling)
- enclosure: `pricesystem/<type>/csv_encloser`


Search Debitor
Welcome, Lena Hartmann!
Default Store View
Search by number, name, postcode or city
Debitor
LUMA
Search entire store here...
What's New
Women
Men
Gear
Training
Sale
PS Industrial
PS Tools
PS Office
PS Electronics
PS Safety
PS Automotive
My Account
My Quotes
Quick Order
My Orders
My Documents
My Ordered Products
My Downloadable Products
My Wish List
Address Book
Account Information
Stored Payment Methods
My Product Reviews
Newsletter Subscriptions
My Sales Staff
My Prices
My Quotes
Request Quote
Compare Products
You have no items to compare.
My Prices
Pricelist Download
Download pricelist Wholesale Core Catalog Download pricelist Enterprise Contract 2026
Product Customer Matrix Download
Download ProductCustomerMatrix Electronics Wholesale Matrix
Select pricetype to see prices
Pricelist
4 Item(s)Show 1 per page
SKU
Product
Qty
Pricelist
Regular price
24-MB04
Strive Shoulder Pack
1
€99.00
€32.00
24-MB01
Joust Duffle Bag
1
€59.00
€34.00
10
€54.00
€34.00
24-MB02
Fusion Backpack
1
€74.00
€59.00
24-MB03
Crown Summit Backpack
1
€89.00
€38.00

Price Adjustment

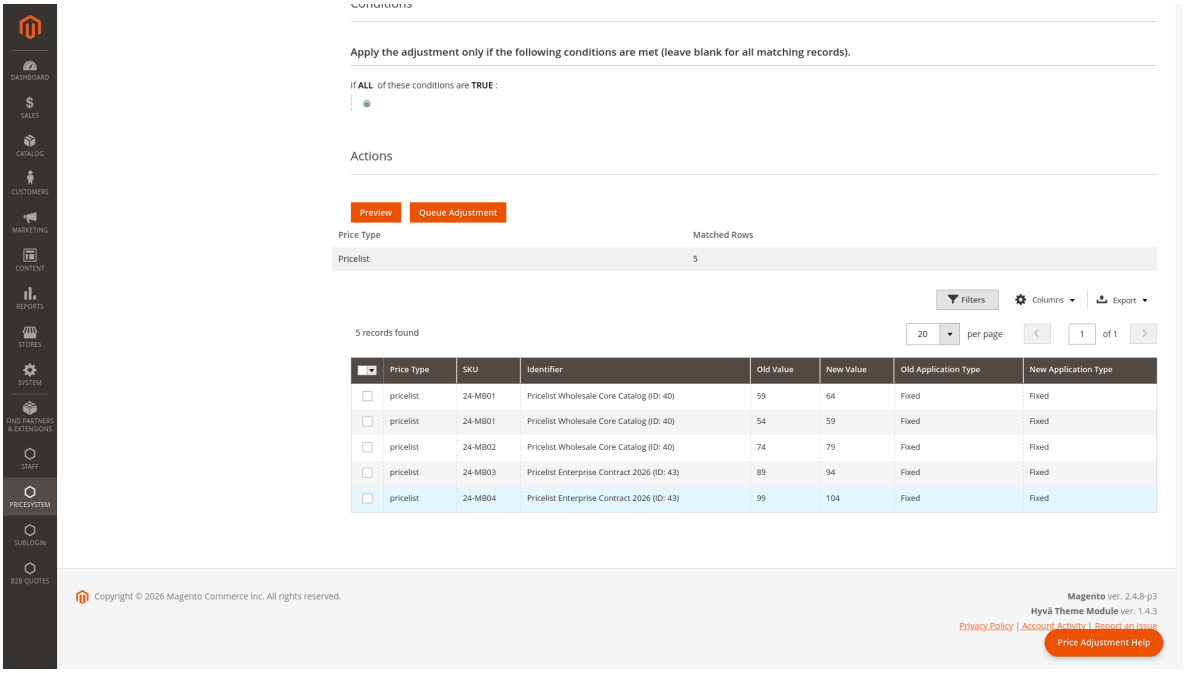
Pricing System Overview · /pricesystem/price-adjustment

The Price Adjustment feature lets you update existing Pricesystem prices in bulk by applying an increase/decrease rule (fixed or percent) with optional filters and conditions.

You generate a preview first, then queue a job; the queued job is applied asynchronously via a Magento message queue consumer.

Quick Links

- How It Works
- Run The Queue Consumer
- Troubleshooting



How It Works

Pricing System Overview · /pricesystem/price-adjustment/how-it-works

Admin UI

Go to:

- Admin -> Pricesystem -> Price Adjustment

The form creates a job with:

- target price types (what you want to adjust)
- optional filter by existing price application types
- new price application type (optional override)
- adjustment direction and mode
- adjustment value
- optional new date range (both from/to must be set to create new dated rows; otherwise existing rows are updated)
- optional website scope (applies to customer_price, categoryprice, and product_customer_matrix targets)
- optional conditions (Magento rule conditions)

What Price Types Can Be Adjusted

The current UI exposes these target price type codes:

- `customer_discount`
- `group_discount`
- `customer_price`
- `categoryprice`
- `pricelist` and `base_pricelist`
- `product_customer_matrix`

Async Execution (Queue)

On submit:

1. A record is created in `pricesystem_price_adjustment_job` with status `queued`.
2. A message is published to the topic `mageb2b.pricesystem.price_adjustment` (payload includes `job_id`).
3. The consumer `mageb2b.pricesystem.price_adjustment.consumer` processes the job:
    - sets job status to `processing`
    - applies updates in a database transaction
    - updates progress periodically
    - sets status to `completed` (or `failed`)

Date Range Conflicts

If a job creates a new date range (from/to) for price rows and it would overlap with existing rows, the job can skip those rows and write conflict messages into the job "message" field.

Database Tables

- Jobs: `pricesystem_price_adjustment_job`
- The job then updates the underlying price tables (customerprice/categoryprice/pricelist/matrix, etc.) depending on the selected target types.

Run The Queue Consumer

Pricing System Overview · /pricesystem/price-adjustment/run-the-consumer

Price Adjustment jobs are processed by this consumer:

- Consumer name: `mageb2b.pricesystem.price_adjustment.consumer`
- Queue: `mageb2b.pricesystem.price_adjustment.queue`
- Connection: `db`

Start The Consumer

Run on your Magento server:

```
php bin/magento queue:consumers:start
mageb2b.pricesystem.price_adjustment.consumer
```

Monitor Job Status

In the admin UI, jobs move through:

- `queued` -> `processing` -> `completed`

If a job fails, it is marked as:

- `failed`

and the error message is stored on the job.

Notes For Production

- Run the consumer as a supervised process (systemd/supervisord/k8s).
- If you do not run the consumer, jobs will stay in `queued` state and no prices are updated.

Troubleshooting

Pricing System Overview · /pricesystem/price-adjustment/troubleshooting

Job Stays On "Queued"

Most common cause:

- the consumer is not running

Start it:

```
php bin/magento queue:consumers:start
mageb2b.pricesystem.price_adjustment.consumer
```

Job Fails

Check:

- the job "message" field in admin (or in `pricesystem_price_adjustment_job.message`)
- Magento logs (`var/log/system.log`, `var/log/exception.log`)
- queue/cron setup if you run consumers via cron

Some Rows Were Not Updated (Date Range Conflicts)

If you set a new `date_from` and `date_to`, Pricesystem validates date overlaps for certain price tables.

When a conflict is detected:

- the row is skipped
- a conflict message is added to the job message summary

Price Selection

Pricing System Overview · /pricesystem/price-selection

Pricesystem calculates multiple candidate prices for a product (from Magento core prices and from installed Pricesystem price types) and then selects the final purchasable price using a configurable strategy.

This section explains:

- which price selection strategies exist (Lowest / Highest / Sort order / Formula)
- how customer and customer-group overrides work (Advanced Config Add-On)
- how strike-through price can be taken from the "next available" price (Sort order)

Quick Links

- System Configuration (Global)
- Sort Order Strategy (Priority List)
- Customer Group Overrides
- Customer Overrides + Fallback
- Formula Pricing

What Is A "Price Type"?

Internally, Pricesystem works with price type codes (also called "price codes"), for example:

- Magento core: `tier_price` (group price), `special_price`, `catalog_rule_price`, `orig_price`
- Pricesystem: `customer_discount`, `group_discount`
- Optional price type modules: `customer_price`, `categoryprice`, `pricelist`, `base_pricelist`, `product_customer_matrix`

The final price strategy selects one of these candidate price types based on your configuration.

Customer Group Overrides

Pricing System Overview · /pricesystem/price-selection/customer-group-overrides

Customer group overrides require the Pricesystem Advanced Config Add-On:

- Composer package: `mageb2b/pricesystem-advancedconfig`
- Magento module: `MageB2B_PricesystemCoreAdvancedConfig`

Where To Configure

- Admin -> Customers -> Customer Groups -> Edit

Price Select Rule (Group)

On group level you can override the global price selection strategy (depending on the Advanced Config UI and version).

Important behavior:

- Group-level selection is used for **guests** (NOT_LOGGED_IN group).
- For logged-in customers, group-level selection is only used if the customer-level rule is set to legacy fallback value `priceselect=3` (not exposed as a selectable UI option in current source).

Group vs System Configuration

If the group-level rule is set to "Default/System Config", the global system configuration is used.

Group Custom Sort Order

If the group-level rule is set to "Custom sort order":

- Pricesystem uses the group-specific priority list stored in `pricesystem_customer_group_sort_position`
- If the group has no custom list configured, it falls back to the global `price_sort_order`

Group Formula

If formula pricing is active (strategy = formula), the formula itself is resolved in this order:

1. Customer formula (if set)
2. Group formula (if set)
3. System configuration formula

Details: Formula Pricing

Customer Overrides

Pricing System Overview · /pricesystem/price-selection/customer-overrides

Customer overrides require the Pricesystem Advanced Config Add-On:

- Composer package: `mageb2b/pricesystem-advancedconfig`
- Magento module: `MageB2B_PricesystemCoreAdvancedConfig`

Where To Configure

- Admin -> Customers -> All Customers -> Edit

Customer Price Select Rule

The customer-level setting can override the system configuration.

Selectable options in the current UI:

Default

Uses the default customer behavior from configuration.

Lowest

Uses the lowest-price strategy.

Highest

Uses the highest-price strategy.

Price select from system config

Uses the global strategy from:

- `pricesystem/price_select_rule/priceselect`

Custom sort order

Uses a customer-specific priority list from `pricesystem_customer_sort_position`.

If the customer has no custom list defined, it falls back to the global sort order list. If customer-specific sort rows exist, sort-order strategy is enforced for that customer.

Individual price formula

Selects the formula pricing strategy. The formula itself is resolved in this order:

1. Customer formula (if set)
2. Group formula (if set)
3. System configuration formula

Default Setting Note

Advanced Config provides this config path:

- Config path: `pricesystem/advanced/default_priceselect`

Important: in the current code, this value is not automatically applied for new customers in the save flow.

Legacy note: value `3` (fallback to group) exists in code but is not exposed as a selectable UI option in current source.

Formula Pricing

Pricing System Overview · /pricesystem/price-selection/formula-pricing

Formula pricing requires the Pricesystem Advanced Config Add-On:

- Composer package: `mageb2b/pricesystem-advancedconfig`
- Magento module: `MageB2B_PricesystemCoreAdvancedConfig`

When Formula Pricing Is Used

Formula pricing is used when the selected strategy is "Individual price formula":

- globally via `pricesystem/price_select_rule/priceselect = 4`, or
- via customer/group override (customer/group UI option "Individual price formula")

Where To Set The Formula

The formula itself is resolved in this order:

1. Customer attribute `price_formula` (if set and non-empty)
2. Customer group column `price_formula` (if set and non-empty)
3. System configuration `pricesystem/price_select_rule/formula`

Variables (Price Codes)

In formulas you reference price codes such as:

- `customer_price`
- `pricelist`
- `special_price`
- `tier_price`
- `orig_price`

Notes:

- Both `code_with_underscores` and `codewithunderscores` are accepted.
- `orig_price` is always available as a reference for fallback and for formulas.
- If `pricesystem/price_calculation/skip_zero_price = Yes`, zero or negative prices are ignored for formula inputs.

Supported Functions

The formula engine supports helpers like:

- `low(code)` (minimum)
- `high(code)` (maximum)
- `avg(code)` (average)

and utility functions:

- `min(a, b, ...)`
- `max(a, b, ...)`

- `round(value, decimals?)`
- `abs(value)`

Missing Input Behavior

Configure how missing values are handled:

- Config path: `pricesystem/price_select_rule/formula_missing_input_behavior`

Typical behaviors:

- Fail and use fallback strategy (strict)
- Treat missing values as `0` (lenient)

Formula Error Fallback

If the formula is empty, invalid, or evaluates to a non-positive result, Pricesystem can fall back to a strategy:

- Config path: `pricesystem/price_select_rule/formula_fallback_strategy`

Fallback strategies include:

- Lowest
- Highest
- Sort order
- Original price

Sort Order Strategy

Pricing System Overview · /pricesystem/price-selection/sort-order-strategy

When the global "Price Select Rule" is set to **Sort order**, Pricesystem selects the first available price from your configured priority list.

How The Selection Works

1. Pricesystem builds a map of candidate prices by price code.
2. It iterates the configured `price_sort_order` list from top to bottom.
3. The first price code that exists and has a valid value is selected.
4. If nothing matches, it falls back to `orig_price` (Magento original price).

Skip Zero Price

If `pricesystem/price_calculation/skip_zero_price = Yes`, any price that resolves to `0.00` (or lower) is treated as "not set" and skipped.

Strike-Through: "Next Available Price"

If `pricesystem/price_select_rule/use_next_available_price_in_strike_through = Yes`, Pricesystem can show the strike-through price as:

- the next available price in your priority list

Example:

Configured order:

1. `special_price`
2. `pricelist`
3. `orig_price`

If `special_price` is selected as the final price, the strike-through price can become `pricelist` (the next available), instead of the original Magento price.

Skip Discounts For The Strike-Through Price

If `pricesystem/price_select_rule/skip_customer_group_discounts_for_next_available_price = Yes`, the strike-through price is evaluated without applying additional customer/group discounts.

System Configuration (Global)

Pricing System Overview · /pricesystem/price-selection/system-configuration

Configure Pricesystem here:

- Admin -> Stores -> Configuration -> MageB2B -> Pricesystem

Enable Pricesystem (Per Website)

- Config path: `pricesystem/general/active`
- Scope: Website

If Pricesystem is disabled for a website, Magento's default pricing is used.

Strategy: Price Select Rule

This decides how Pricesystem selects the final price when multiple candidate prices exist.

- Config path: `pricesystem/price_select_rule/priceselect`
- Values:
  - `1` = Lowest
  - `2` = Highest
  - `3` = Sort order
  - `4` = Individual price formula (requires Pricesystem Advanced Config Add-On)

Sort Order List (Priority Order)

Used only when `priceselect = Sort order`.

- Config path: `pricesystem/price_select_rule/price_sort_order`
- Meaning: Top = highest priority (first match wins)

Example (conceptual):

1. `customer_price`
2. `pricelist`
3. `special_price`
4. `tier_price`
5. `orig_price`

Details: Sort Order Strategy

Optional: Next Available Price As Strike-Through

Used only when `priceselect = Sort order`.

- Config path: `pricesystem/price_select_rule/use_next_available_price_in_strike_through`
- If enabled, the strike-through "regular price" can be the next available price from the sort order list (instead of always using the original Magento price).

There is also a related setting:

- Config path: `pricesystem/price_select_rule/skip_customer_group_discounts_for_next_available_price`
- If enabled, the next-available strike-through price is shown without additional customer/group discounts applied.

Notes About Zero Prices

- Config path: `pricesystem/price_calculation/skip_zero_price`
- If enabled, candidate prices `<= 0` are ignored by the selection strategies that support it.

Troubleshooting

Pricing System Overview · /pricesystem/troubleshooting

Common issues and solutions for the Pricing System.

Price Not Displaying

Symptoms

- Customer sees catalog price instead of custom price
- "My Prices" shows empty

Solutions

1. **Check customer is logged in**
    - Customer-specific entries require login
    - Group/context-based pricing can still apply for guests
2. **Verify date validity**
    - Check from_date and to_date
    - Ensure current date is within range
3. **Check website scope**
    - Verify website_id matches current store
    - Use 0 for all websites
4. **Review priority settings**
    - Higher priority may be overriding
    - Check price selection strategy
5. **Clear caches**

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

Wrong Price Calculation

Symptoms

- Price is incorrect
- Wrong tier applied

Solutions

1. **Enable debug logging**

    - **Stores > Config > Pricesystem > Debug > Enable log**: Yes
    - Check `var/log/pricesystem.log`

2. **Check quantity tiers**
    - Verify cart qty meets tier threshold
    - Review tier configuration

3. **Review priority chain**
    - Check all matching rules
    - Verify priority values

4. **Test in incognito**
    - Rule out browser caching

Import Failures

Symptoms

- CSV import fails
- Validation errors

Solutions

1. **Check CSV format**
    - UTF-8 encoding
    - Correct delimiter (default `,`; match your import command/settings if overridden)
    - All required columns present

2. **Verify data integrity**
    - Valid product SKUs
    - Valid customer emails
    - Valid category IDs

3. **Check for duplicates**
    - Unique constraint violations
    - Remove duplicate rows

4. **Review error log**

```
tail -100 var/log/system.log
tail -100 var/log/exception.log
```

Performance Issues

Symptoms

- Slow page loads

- High database load

Solutions

1. **Check cache status**

   ```
   php bin/magento cache:status
   ```

2. **Optimize cache lifetime**
   - Increase cache TTL if prices change infrequently
   - **Stores > Config > Pricesystem > Price Cache Lifetime**

3. **Index optimization**
   - Ensure database indexes are present
   - Run reindex if needed

4. **Review price rules count**
   - Large numbers of rules impact performance
   - Consider consolidating rules

API Errors

401 Unauthorized

- Verify API credentials
- Check token expiration
- Confirm permission model for the endpoint type (integration ACL vs customer token vs anonymous context routes)

404 Not Found

- Verify endpoint URL
- Check API module is installed
- Confirm resource exists

500 Server Error

- Check `var/log/exception.log`
- Verify request payload format
- Check database connectivity

Common Error Messages

"Price not found for customer"

Customer has no assigned prices:

- Create customer price entries
- Assign customer to pricelist

- Check customer group pricing

"Duplicate entry"

Unique constraint violation:

- Check for existing price with same parameters
- Use UPDATE instead of INSERT
- Review (customer, product, qty, website, dates) combination

"Invalid date format"

Date parsing failed:

- Use YYYY-MM-DD format
- Ensure dates are valid
- Check locale settings

Debug Checklist

When troubleshooting price issues:

1. ☐ Correct customer context is used (logged-in customer for customer-specific prices, or guest/NOT_LOGGED_IN context where applicable)
2. ☐ Price entry exists in database
3. ☐ Date range is valid (or null)
4. ☐ Website scope matches
5. ☐ Quantity threshold is met
6. ☐ Priority doesn't conflict
7. ☐ Cache is cleared
8. ☐ Debug log is checked

Getting Help

If issues persist:

1. Collect debug logs
2. Note Magento and extension versions
3. Document steps to reproduce
4. Contact support at info@softwaresilo.io