Customer Specific Prices 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 Customer Specific Prices

Table of Contents

Customer Specific Prices Documentation · customerprice

Overview
Customer Specific Prices
Getting Started
Installation
Configuration
Create a customer price
Admin Permissions (ACL)
Features
Website scope
Quantity thresholds
Date ranges
Cart price rule condition
Addons
Advanced Config
CSV import and export
REST API
GraphQL
Sample data
Reference
Configuration Fields
Database Reference
Troubleshooting
Common Issues
FAQ

Customer Specific Prices

Customer-specific prices let you agree a product price with one customer without changing the catalog price for everyone else. A record can apply to one website, start and end on chosen dates, and use quantity thresholds for volume pricing.

Pricesystem Core is included with the product. It provides the shared price calculation and decides how a customer price competes with Magento prices and other installed Pricesystem modules.

When to use customer-specific prices

Use this module when the agreement names both the customer and the product. Typical examples include a contracted unit price, a temporary project price, or different rates at quantity thresholds.

Choose a different Pricesystem module when the agreement has a broader shape:

Requirement	Module
One price for one customer and product	Customer-specific prices
A reusable list assigned to customers	Price Lists
Pricing by category	Category Price
Many customer and product combinations managed as a matrix	Product-Customer Matrix

What a price record controls

A customer price contains the customer, product, minimum quantity, value, application type and website. Dates are optional. Fixed prices replace the selected base price. Surcharges and discounts calculate from the chosen base price.

Customer prices only apply to **signed-in customers** — guests always see normal catalog prices, since there is no account to attach the agreement to.

For quantity pricing, the module uses the closest eligible threshold. If the order quantity is 24 and records exist for quantities 1, 10 and 50, the quantity 10 record applies.

The module also adds the cart price rule condition **Customerprice > Customer has customerprice** (Yes/No) under *Cart Price Rules* > *Conditions*. Use it to give a discount to every customer who has at least one active customer price — for example a small extra rebate for contract customers.

Start here

1. Install the product.

2. Review the configuration.
3. Create a customer price.
4. Check quantity thresholds, date ranges and website scope if the agreement needs them.
5. Read price selection if several price types can match the same product.
6. Restrict admin access with ACL permissions.

Customers can also receive a CSV list of their current prices through My Prices.

Optional packages

- Advanced Config adds customer and customer-group selection rules.
- CSV Import/Export handles larger price sets through Magento's data transfer tools.
- REST API provides read and write endpoints.
- GraphQL provides storefront-oriented price queries.
- Sample Data creates example records for development environments.

For common questions, see the FAQ and troubleshooting guide.

View the product page

Installation

Pricesystem Core is included with the product. Composer installs it as a dependency, so you do not need to buy or install the core separately.

Install the package

Run these commands from the Magento project directory. Replace `repo.softwaresilo.io`, `softwaresilo`, `https://repo.softwaresilo.io/` and `<token>` with the values shown on the license card under **Account > Licenses** — see Installation & Updates.

```
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
php bin/magento setup:upgrade
php bin/magento setup:di:compile
php bin/magento setup:static-content:deploy
php bin/magento cache:flush
```

Composer may already have enabled the module and its dependencies. The explicit `module:enable` command is safe and makes the intended state clear.

Check the installation

```
php bin/magento module:status MageB2B_PricesystemCustomerprice
```

Then sign in to the Magento Admin and open **Pricesystem > Customerprice > All Customerprices**. The grid should load and show an **Add New Customerprice** action.

Continue with configuration before entering production prices.

Configuration

Open **Stores > Configuration > MageB2B > Pricesystem**. The Customerprice section controls the CSV download in the customer account. Shared calculation settings are documented under Pricesystem configuration.

Customer price settings

Setting	What it does	Default
Enable download link	Adds the personal price CSV to the signed-in customer's account	Yes
CSV delimiter	Separates columns in the generated file	Semicolon (`;`)
CSV encloser	Wraps text fields that need quoting	Double quote (`"`)

Use a delimiter that matches the spreadsheet software used by your customers. A semicolon is often easier in locales where a comma is the decimal separator.

Shared price selection

Pricesystem Core can select the lowest price, highest price, or the first available price in a configured priority list. If a customer-specific price should take precedence, place `customer_price` before the competing price codes in the priority list.

Read how price selection works before changing this setting on a live store. A global change can affect every installed Pricesystem price type.

After saving the configuration, clear the Magento cache and test with a customer account that has one known customer price.

Create a customer price

Open **Pricesystem > Customerprice > All Customerprices** and select **Add New Customerprice**. You can also manage a customer's records from the customer edit screen.

Complete the price record

Field	Purpose
Customer	The customer who receives the price
Product	The product to which the agreement applies
Quantity	The minimum quantity for this price tier
Price	The fixed amount, surcharge or discount
Price Application Type	Fixed, Surcharge, Surcharge %, Discount or Discount %
Base Price	The product price used for a relative surcharge or discount
Website	All websites or one specific website
From Date / To Date	Optional date range in which the record is active

For a fixed contract price, choose **Fixed** and enter the final unit price. For a percentage discount, choose **Discount %**, enter the percentage as the value, and confirm the base price source.

Example

Northstar Industrial has agreed to pay 86.50 for product `FIELD-BAG-24` from a quantity of ten:

```
Customer: Northstar Industrial
Product: FIELD-BAG-24
Quantity: 10
Price: 86.50
Price Application Type: Fixed
Website: Main Website
From: empty
To: empty
```

Save the record, sign in as the selected customer, and add at least ten units to the cart. Test a quantity below the threshold as well. This confirms both the agreement and the fallback price.

Avoid ambiguous records

Keep quantity thresholds and date ranges deliberate. Two records for the same customer, product and context can be difficult to reason about even when Magento accepts them. If a contract changes, end the old record before the new one begins.

Continue with quantity thresholds, date ranges, or website scope.

Admin Permissions (ACL)

Customer Price adds granular admin permissions so you can give pricing or support users access without full admin rights.

Where To Configure Roles

In Magento Admin go to:

- **System > Permissions > User Roles** (create/edit a role)
- **System > Extensions > Integrations** (if you use integration tokens)

The permissions below show up under **Pricesystem Customerprice All** in the role resource tree.

Admin Menu (What Maps To What)

Admin Menu	Required ACL
`Pricesystem > Customerprice`	`MageB2B_PricesystemCustomerprice::all`
`Pricesystem > Customerprice > All Customerprices`	`MageB2B_PricesystemCustomerprice::manage`

Main ACL Resources

Permission	What It Controls
`MageB2B_PricesystemCustomerprice::manage`	Customer price grid and record screens
`MageB2B_PricesystemCustomerprice::save`	Create/update customer prices
`MageB2B_PricesystemCustomerprice::delete`	Delete customer prices

The module declares no own `config` resource. Access to **Stores > Configuration > MageB2B > Pricesystem** is governed by the shared `MageB2B_PricesystemCore::config` resource.

Tip: Align ACL With REST API Access

If you use the **REST API Add-On** (package `mageb2b/pricesystem-customerprice-api`), the same resources protect its routes:

- Read and search routes require `manage`

- Create, update and bulk-save routes require `save`
- Single, bulk and per-customer deletes require `delete`

Keep the integration user's permissions minimal. See REST API for installation and the endpoint reference.

Website scope

A customer price can apply to all websites or to one Magento website. Use website scope when the same customer has separate commercial terms for different storefronts.

During price calculation, the module considers the current website and global records. When the quantity threshold is the same, a website-specific record is preferred over the global record.

Example

Northstar Industrial has a global unit price of 95.00, but the German website has a negotiated price of 91.00. Create two otherwise matching records:

Website	Qty	Value
All Websites	1	95.00
German Website	1	91.00

The German storefront uses 91.00. Other websites can use the global 95.00 record.

Website scope does not convert currencies. Enter values that make sense for the website's configured currency and test the customer account on each affected storefront.

You can find website names and scopes under **Stores > All Stores**. Use the website selector in the customer price form instead of relying on numeric IDs from another environment.

Quantity thresholds

Each customer price record has a minimum quantity. The module sorts matching records by quantity from highest to lowest and uses the first threshold that does not exceed the requested quantity.

For example:

Minimum quantity	Unit price
1	100.00
10	94.00
50	88.00

An order for 24 units uses 94.00. An order for 50 units uses 88.00. If the order quantity is below every customer-price threshold, this price type has no applicable record and Pricesystem continues with its configured selection logic.

The closest eligible threshold wins. The module does not scan the eligible tiers for the lowest numerical price. This matters if a higher threshold contains a surcharge or an intentionally higher contract rate.

Create tiers

Create one record for each threshold. Keep the customer, product, website and date window consistent when the records belong to the same agreement. Change only the quantity and value.

Test the exact threshold, one unit below it, and a quantity above it. If the displayed price is still unexpected, review price selection because another price type may have won after the customer price was calculated.

Date ranges

The **From** and **To** fields limit when a customer price can be used. Both boundaries are included.

- Leave both fields empty for a price without a date limit.
- Set only **From** when a new agreement starts on a known date and has no planned end.
- Set only **To** when an existing agreement should expire on a known date.
- Set both for a fixed contract or campaign period.

The same date filter applies when the customer downloads current prices. Future and expired records do not appear as active prices.

Replacing an agreement

End the old record on the day before the new record begins. For example, an old price can end on 30 September and its replacement can begin on 1 October. Avoid overlapping records for the same customer, product, quantity and website unless the overlap is intentional and you have tested the selection result.

If a scheduled price does not appear, check the store date, the record's website and the requested quantity before changing the date fields.

Cart price rule condition

Customer Price registers the condition **Customerprice > Customer has customerprice** under *Marketing > Cart Price Rules > Conditions*. It is a Yes/No check on the quote's customer, so a promotion can apply only to accounts that already own at least one customer price.

The condition is met when at least one currently valid customer price record exists for the customer. Records outside their **From**/**To** date range do not count. The check does not look at the cart contents — it only verifies that some record exists.

A typical use is a small extra rebate for contract customers without maintaining a separate customer group. Test the rule with a customer whose only records are expired — they must not receive the discount.

Advanced Config

Advanced Config changes how Pricesystem chooses between available price types. It is useful when customer-specific prices must follow rules that differ by customer or customer group.

The package can provide:

- customer and customer-group overrides for the global selection rule;
- custom priority lists;
- formula-based price selection;
- a fallback path from a customer rule to the customer-group rule and then to the system configuration.

These are shared Pricesystem rules. They do not change the fields of a customer price record.

Install package `mageb2b/pricesystem-advancedconfig`, run Magento's normal setup commands, and then follow the Advanced Config guide. The customer override and customer-group override pages explain the precedence.

CSV import and export

The Import/Export package adds customer prices to Magento's standard data transfer tools. It is intended for contract price lists that are easier to maintain in a spreadsheet or another business system.

Install the package

```
composer require mageb2b/pricesystem-customerprice-
importexport:*
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Prepare a file

Open **System > Data Transfer > Export** and export the customer price entity first. Use that file as the template for the target store. This avoids guessing identifiers and column names.

The customer price import supports these business fields:

Column	Meaning
`customer_id`	Numeric customer ID or account email — both always work on import. With per-website account sharing, the email is resolved within the row's website scope
`product_id`	SKU by default; numeric product ID when **Customerprice Import > Use Product ID as** is set to Product ID
`qty`	Minimum quantity
`price`	Price value
`from_date`, `to_date`	Optional active period
`price_application_type`	Fixed, nominal/percentage surcharge, or nominal/percentage discount
`price_attribute_id`	Base price source for relative calculations
`website_id`	Numeric website ID

`product_id` is a strict switch: the configured alias decides whether the column must contain SKUs or numeric IDs. For exports, **Customerprice Export** controls whether `product_id` and `customer_id` are written as SKU/email (default) or numeric IDs. Configure these aliases under **Stores > Configuration > MageB2B > Pricesystem**

before producing the template.

Import behavior

The admin import and the CLI `--behavior` option accept the standard Magento import behaviors:

Behavior	Effect
`append` / `add_update`	Validate and save the file's rows; existing records stay untouched
`replace`	**Deletes every existing customer price first**, then imports the file
`delete`	Deletes only the records that match rows in the file

`replace` wipes the whole customerprice table before importing — it is not a per-row upsert. Never run a replace import without a current export or backup.

Import safely

1. Export a small known set.
2. Keep the header row unchanged.
3. Add a few records and run the import with validation enabled.
4. Review rejected rows before importing the full file.
5. Test one fixed price, one quantity tier and one dated price in the storefront.

Date-range validation can check rows within the file and against existing customer prices. Keep it enabled unless the import is part of a controlled replacement process.

Command-line import and export

The same package also provides CLI commands for scripted or scheduled transfers:

```
php bin/magento pricesystem:import-customerprice <import_path>
php bin/magento pricesystem:export-customerprice
```

The import accepts the same columns as the admin import. The customer-price grid also offers a standard Magento export (CSV/XML) through its export button, plus a Delete mass action.

Extension points for developers

- The import dispatches the event `pricesystem_customerprice_import_before_save` before each batch is

written. Observers receive the import parameters and the insert/update row arrays.

- The import model's `checkForAdditionalValues($entityRow, $rowData)` method is intentionally public so a custom module can map extra CSV columns into the entity row through an after-plugin.

For the customer-facing CSV, see My Prices. That download is separate from Magento's administrative import and export.

REST API

The API package exposes customer price records through Magento Web API. Create a Magento integration with only the resources required by the connecting system. Do not use an administrator token in a scheduled integration. The routes are protected by the Customer Price ACL resources: `manage` for reads, `save` for creates/updates and `delete` for deletes.

Install the package

```
composer require mageb2b/pricesystem-customerprice-api:*
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Endpoints

Method	Endpoint	Purpose
`GET`	`/V1/pricesystem/customerprice/:id`	Read one record
`GET`	`/V1/pricesystem/customerprice/search`	Search with Magento search criteria
`POST`	`/V1/pricesystem/customerprice`	Create a record
`PUT`	`/V1/pricesystem/customerprice/:id`	Update a record
`POST`	`/V1/pricesystem/customerprice/savebulk`	Save several records
`DELETE`	`/V1/pricesystem/customerprice/:id`	Delete one record
`DELETE`	`/V1/pricesystem/customerprice/deleteByCustomerId/:customerId`	Delete records for a customer
`POST`	`/V1/pricesystem/customerprice/deletebulk`	Delete several records

Magento's search endpoint accepts standard `searchCriteria` query parameters. Start with a narrow filter and a small page size when synchronizing large catalogs.

Before sending a bulk request, validate customer, product, website and quantity

identifiers in the source system. Bulk responses distinguish records that were saved from records that failed, so process the result instead of treating a successful HTTP request as proof that every row was stored.

When the shared **API settings > Use custom API fields for sync** option is enabled (under **Stores > Configuration > MageB2B > Pricesystem**), `customer_id` values in request bodies and the `:customerId` of `deleteByCustomerId` are resolved through the configured customer attribute instead of the numeric customer ID.

The shared Pricesystem REST guide covers price calculation endpoints. Those endpoints return calculated prices; the endpoints above manage customer price records.

GraphQL

The GraphQL package adds queries for individual customer price records and paginated lists.

Install the package

```
composer require mageb2b/pricesystem-customerprice-graph-ql:*
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Queries

Query	Use	Token
`customerprice(id: Int!)`	Read one record by ID	Admin or integration
`customerprices(filter, pageSize, currentPage)`	Search records available to an authorized caller	Admin or integration
`myCustomerprices(filter, pageSize, currentPage)`	Read prices for the signed-in customer	Customer

`customerprice` and `customerprices` reject customer and guest calls with an authorization error — they are record-management queries, not storefront queries.

Use a customer token for `myCustomerprices`. The query derives the customer from the authenticated session rather than accepting another customer's ID. Note that `entity_id` is the product ID — the schema does not expose a `product_id` field.

```
query MyCustomerPrices {
  myCustomerprices(pageSize: 20, currentPage: 1) {
    total_count
    items {
      id
      customer_id
      entity_id
      qty
      value
      website_id
    }
  }
}
```

Request only the fields the storefront needs and paginate list queries. GraphQL reads records; use the REST API or Magento Import/Export for write workflows.

Sample data

The sample-data packages create example customers, products and customer price records for a development or demonstration environment. Do not install them in a production store.

```
composer require mageb2b/pricesystem-sample-data:*
composer require mageb2b/pricesystem-customerprice-sample-data:*
php bin/magento setup:upgrade
php bin/magento cache:flush
```

Check the installed customer price fixtures with:

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

Reset only the customer price sample set with:

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

The reset command is destructive for records managed by this sample package. Use it only in a disposable environment after checking what data the command identifies.

For the shared Pricesystem fixtures and commands, see Pricesystem sample data.

Configuration Fields

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

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-customerprice`

Customerprice settings

Field	Configuration path	Input	Scope
Enable download link	`pricesystem/customerprice/enable_download`	Select	Default, Website
Csv Delimeter	`pricesystem/customerprice/csv_delimeter`	Text	Default, Website
Csv Encloser	`pricesystem/customerprice/csv_encloser`	Text	Default, Website

`mageb2b/pricesystem-customerprice-importexport`

Customerprice Import

Field	Configuration path	Input	Scope
Use Product ID as	`pricesystem/customerprice_import/product_id_alias`	Select	Default, Website
Enable Date Range Validation	`pricesystem/customerprice_import/enable_date_range_validation`	Select	Default, Website
Enable Date Range Validation Between Importing And Existing	`pricesystem/customerprice_import/enable_date_range_validation_existing`	Select	Default, Website

Customerprice Export

Field	Configuration path	Input	Scope
Use Product ID as	`pricesystem/customerprice_export/product_id_alias`	Select	Default, Website
Use Customer ID as	`pricesystem/customerprice_export/customer_id_alias`	Select	Default, Website

Database Reference

Each Customer Price record connects one customer, one product and one website. This is the core relationship to consider when importing negotiated prices from an ERP or pricing service.

Relationship	Why it matters
`pricesystem_customerprice` applies to one `customer_entity`	Identifies the customer who receives the negotiated price.
`pricesystem_customerprice` prices one `catalog_product_entity`	Identifies the Magento product covered by the price.
`pricesystem_customerprice` belongs to one `store_website`	Keeps customer prices separate between websites.
`pricesystem_customerprice` can use one `eav_attribute`	Allows an alternative product price attribute to be used where configured.

Use Magento Admin, the import add-on or the module API for updates. Direct database writes bypass validation and cache handling.

Common Issues

A customer price does not appear

Check the record in this order:

1. The storefront customer is the customer selected in the record.
2. The product matches the product in the cart.
3. The requested quantity meets the minimum quantity.
4. The record is global or belongs to the current website.
5. The current date is within the optional date range.
6. Pricesystem is enabled for the website.

If the customer price is available but the final price differs, inspect the shared price selection rule.

The wrong quantity tier applies

The closest eligible threshold wins. For a quantity of 24, thresholds 1 and 10 are eligible, while 50 is not. The quantity 10 record applies even if its numerical price is higher than the quantity 1 record.

Check for a website-specific record at the same threshold. It is preferred over the global record.

A scheduled price starts or ends unexpectedly

Both date boundaries are included. Confirm the store date and look for overlapping records with the same customer, product, quantity and website. End an old agreement before its replacement starts.

The customer cannot download prices

Confirm that **Enable download link** is active, the customer is signed in, and current records exist for that customer and website. Future and expired records are not included as active prices.

If the file opens in one column, change the configured CSV delimiter to match the spreadsheet application's import settings.

Import rows are rejected

Export a fresh template from the same Magento environment. Check the configured product and customer aliases, then verify customer, SKU, website, quantity, application type and date values. Keep date-range validation enabled while diagnosing the file.

Changes are not visible after saving

Flush Magento cache and retest with a clean customer session. If Pricesystem price caching is enabled, the cache may still contain the previous calculation until it is cleared or expires.

FAQ

Does Pricesystem Core need a separate purchase?

No. The shared core package is included with Customer-specific prices and is installed as a dependency.

Which price applies when several quantity tiers match?

The record with the highest minimum quantity that does not exceed the requested quantity applies. The module does not choose the cheapest eligible tier.

Can a price be limited to one website?

Yes. Select a website in the price record. A record scoped to **All Websites** acts as the global option; a matching website-specific record is preferred at the same quantity threshold.

Can I schedule a contract price?

Yes. Use the optional **From** and **To** dates. Both date boundaries are included. Leave either side empty for an open-ended period.

Can I enter a discount instead of a final price?

Yes. Application types include fixed price, nominal or percentage surcharge, and nominal or percentage discount. For relative calculations, confirm which product price attribute is used as the base.

Why does another price win in the storefront?

Customer Price supplies one candidate price to Pricesystem. The shared selection rule then compares it with other available types such as special price, tier price or price-list price. Review price selection and the position of `customer_price`.

Can customers download their agreed prices?

Yes. Enable the download link in configuration. The signed-in customer receives current, applicable customer price records as CSV. The delimiter and enclosure are configurable.

How should I maintain thousands of records?

Use the CSV Import/Export package or the REST API. Export a template first and test a small batch before running a full synchronization.

Is GraphQL available for storefronts?

Yes, through the optional GraphQL package. The `myCustomerprices` query uses the signed-in customer context.

When should I use Price Lists instead?

Use Price Lists when a reusable set of prices should be assigned to one or more customers. Use Customer-specific prices when the agreement directly names a customer and product.