Customer Specific Prices

PDF Manual

MODULE

customerprice

LAST UPDATED

2026-03-12

Table of Contents

Customer Specific Prices · customerprice

- Overview
- Customer Specific Prices
- Getting Started
  - Installation & Setup
  - Creating Customer Prices
- Features
  - Multi-Website Configuration
  - Quantity-Based Pricing Tiers
  - Time-Based Pricing
- Addons
  - Advanced Config
  - CSV Import/Export
  - SOAP / REST API
  - Sample Data
- Price Selection
  - Customer Group Overrides
  - Customer Overrides + Fallback
  - Formula Pricing
  - Sort Order Strategy
  - System Configuration (Global)

Customer Specific Prices

Customer Specific Prices · /customerprice

Overview

The **Customer Specific Prices Extension** enables B2B merchants to assign individual product prices to specific customers, replacing standard catalog prices with negotiated or contract-based pricing. This powerful tool is essential for businesses that maintain different pricing agreements with various customers based on volume commitments, partnership levels, or special contracts.

Key Features

Individual Customer Pricing

Assign unique prices for any product to any customer. Each customer-product combination can have its own price, completely independent from catalog prices or customer group discounts.

Quantity-Based Tiering

Define different price points based on order quantity for each customer-product combination. Set up graduated pricing where customers receive better rates when ordering larger quantities.

Time-Based Pricing

Configure temporary prices with precise date ranges using from_date and to_date fields. Schedule promotional prices, seasonal adjustments, or contract renewals in advance.

Multi-Website Support

Manage different customer prices across multiple websites within your Magento installation. Assign website-specific pricing using the website_id field with foreign key constraints ensuring data integrity.

Customer Price Download

Allow customers to download their personal pricing information as CSV files directly from their account dashboard. Customers see only their own negotiated prices in an exportable format.

Admin Management Interface

Manage all customer prices through a comprehensive admin grid with filtering, sorting, and mass actions. Create, edit, and delete customer price assignments individually or in bulk.

What You Can Do

- Assign negotiated prices to individual customers
- Create quantity-based pricing tiers
- Schedule time-based pricing campaigns

- Set different prices per website/region
- Allow customers to export their prices
- Bulk import/export with add-on
- API access with add-on

Use Cases

Negotiated Contract Pricing

Implement custom pricing agreements negotiated with individual customers. Store contract-specific prices that override catalog prices entirely.

Volume Commitment Pricing

Reward customers who commit to purchasing specific volumes with graduated pricing tiers. Configure multiple quantity breakpoints where prices decrease as order size increases.

Seasonal Campaign Pricing

Schedule temporary price adjustments for specific customers during promotional periods or seasonal campaigns. Set precise date ranges that automatically activate and deactivate.

Regional Pricing Strategies

Implement different pricing for the same customers across multiple websites or regions. Maintain separate price lists for EU, US, and APAC storefronts centrally.

VIP Customer Benefits

Provide exclusive pricing to VIP or preferred customers as a loyalty benefit. Assign preferential rates on specific products or entire product lines.

Contract Renewal Management

Manage pricing transitions during contract renewals by scheduling new prices with future start dates. Overlap old and new pricing temporarily during transition periods.

Technical Details

Extension: MageB2B_PricesystemCustomerprice **Dependencies:** MageB2B_PricesystemCore >= 2.3.0 **Database Table:** pricesystem_customerprice **Price Code:** customer_price

Getting Started

1. **Installation** - Install and configure the extension
2. **Creating Prices** - Add your first customer-specific prices
3. **Quantity Tiers** - Set up graduated pricing
4. **Time-Based Pricing** - Schedule temporary prices
5. **Multi-Website** - Configure regional pricing
6. **Price Selection** - How Pricesystem selects the final price
7. **My Prices** - Customer account price overview + CSV download

Add-Ons

Extend functionality with these add-ons:

- **SOAP / REST API Add-On** - CRUD WebAPI endpoints
- **CSV Import/Export Add-On** - Bulk import/export via CSV
- **Advanced Config Add-On** - Price selection overrides, custom sort order, formula pricing
- **Sample Data** - Demo entities and example prices

View Extension Details

Installation & Setup

Customer Specific Prices · /customerprice/getting-started/installation

Requirements

- Magento 2.4+
- PHP 8.0+
- MageB2B_PricesystemCore >= 2.3.0

Installation Steps

Install the module via Composer:

```
# Add authentication to composer
composer config bearer.repo.softwaresilo.io <token>
composer config repositories.softwaresilo composer
https://repo.softwaresilo.io/

# Install the module
composer require mageb2b/pricesystem-customerprice:*

# Enable the module
php bin/magento module:enable MageB2B_PricesystemCustomerprice

# Run setup upgrade
php bin/magento setup:upgrade

# Compile DI
php bin/magento setup:di:compile

# Deploy static content
php bin/magento setup:static-content:deploy

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

Configuration

Navigate to: **Stores > Configuration > Pricesystem > Customerprice Settings**

Available Settings

Enable Download Link

Allow customers to download their personal prices as CSV from their account.

- **Path:** `pricesystem/customerprice/enable_download`
- **Type:** Yes/No
- **Default:** No

When enabled, customers will see a "Download My Prices" link in their account dashboard.

CSV Delimiter

Configure the delimiter used in CSV exports.

- **Path:** `pricesystem/customerprice/csv_delimeter`
- **Type:** Text
- **Default:** `,` (comma)
- **Examples:** `;` (semicolon), `|` (pipe), `\t` (tab)

CSV Encloser

Configure the enclosure character used in CSV exports.

- **Path:** `pricesystem/customerprice/csv_encloser`
- **Type:** Text
- **Default:** `"` (double quote)

Verification

After installation, verify the module is active:

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

Expected output:

```
Module is enabled
```

Admin Access

Navigate to: **Catalog > Customer Prices**

You should see the customer price management grid.

Next Steps

- Creating Your First Customer Price
- Setting Up Quantity Tiers
- Configuring Time-Based Pricing

Creating Customer Prices

Customer Specific Prices · /customerprice/getting-started/creating-prices

Overview

Learn how to create your first customer-specific price assignments in the admin panel.

Step-by-Step Guide

1. Navigate to Customer Prices

Go to: **Catalog > Customer Prices**

2. Click "Add New"

Click the "Add New" button in the top right corner.

3. Fill in Required Fields

Customer

Select the customer from the dropdown. You can search by name or email.

Required: Yes

Product

Search and select the product. Enter product name, SKU, or ID.

Required: Yes **Note:** Only simple and virtual products are supported. For configurable products, set prices on child products.

Quantity

The minimum quantity for this price tier.

Required: Yes **Default:** 1

Price Value

The customer-specific price for this product.

Required: Yes **Format:** Decimal (e.g., 99.99)

Website

Select which website this price applies to.

Options:

- **All Websites** (ID: 0) - Price applies globally
- **Specific Website** - Price applies only to selected website

Default: 0 (All Websites)

4. Optional Fields

From Date

Start date for time-based pricing. Leave empty for immediate activation.

Format: YYYY-MM-DD

To Date

End date for time-based pricing. Leave empty for no expiration.

Format: YYYY-MM-DD

Price Application Type

How the price should be applied (configured in Pricesystem Core).

Default: 1 (Replace)

Price Attribute ID

Optional custom price attribute reference for advanced configurations.

5. Save

Click "Save" to create the customer price.

Example: Basic Customer Price

Scenario: Customer "ACME Corp" gets $95 for Product "Widget ABC" (normally $100)

```
Customer: ACME Corp (customer_id: 123)
Product: Widget ABC (SKU: WGT-ABC)
Quantity: 1
Price: 95.00
Website: All Websites
From Date: (empty)
To Date: (empty)
```

Example: Quantity Tier Pricing

Scenario: Customer "Beta Ltd" gets volume discounts

Create multiple entries:

```
Entry 1:
- Customer: Beta Ltd
- Product: Widget ABC
- Quantity: 1
- Price: 100.00

Entry 2:
```

```
- Customer: Beta Ltd
- Product: Widget ABC
- Quantity: 10
- Price: 95.00

Entry 3:
- Customer: Beta Ltd
- Product: Widget ABC
- Quantity: 50
- Price: 90.00
```

The system automatically applies the best price based on cart quantity.

Example: Promotional Price

Scenario: Q1 2025 promotion for specific customer

```
Customer: Gamma Inc
Product: Widget ABC
Quantity: 1
Price: 85.00
From Date: 2025-01-01
To Date: 2025-03-31
Website: All Websites
```

Price activates automatically on January 1st and expires on March 31st.

Validation

The system validates:

- **Uniqueness:** No duplicate entries with same customer + product + qty + dates + website
- **Customer Exists:** Customer ID must be valid
- **Product Exists:** Product ID must be valid
- **Price Format:** Must be valid decimal
- **Date Logic:** to_date must be after from_date (if both set)

Next Steps

- Setting Up Quantity Tiers
- Time-Based Pricing
- Multi-Website Configuration

Multi-Website Configuration

Customer Specific Prices · /customerprice/features/multi-website

Overview

Manage different customer prices across multiple websites within your Magento installation. Set region-specific pricing, handle multi-currency scenarios, and maintain separate price lists for different storefronts.

How It Works

The `website_id` field determines which website(s) a price applies to:

- **website_id = 0**: Price applies to all websites (global)
- **website_id = 1, 2, 3, etc.**: Price applies only to that specific website

When resolving prices, the system:

1. Checks for prices with matching website_id
2. Falls back to website_id = 0 if no website-specific price exists
3. Applies standard catalog price if no customer price matches

Website ID Reference

Finding Website IDs

Admin Method:

Navigate to: **Stores > All Stores**

View your website structure:

```
Main Website (ID: 1)
  └─ Main Store
     └─ English Store View
     └─ German Store View

EU Website (ID: 2)
  └─ EU Store
     └─ EU English View
     └─ EU German View

US Website (ID: 3)
  └─ US Store
     └─ US English View
```

Database Query:

```
SELECT website_id, code, name FROM store_website;
```

Example output:

```
| website_id | code   | name         |
|------------|--------|--------------|
| 0          | admin  | Admin        |
| 1          | base   | Main Website |
| 2          | eu     | EU Website   |
| 3          | us     | US Website   |
```

Use Cases

Regional Pricing Strategy

Scenario: Different prices for US and EU customers

```
US Website (website_id=3):
Customer: ACME Corp
Product: Widget ABC
Quantity: 1
Price: $100.00
Website: US Website

EU Website (website_id=2):
Customer: ACME Corp (same customer)
Product: Widget ABC
Quantity: 1
Price: €90.00
Website: EU Website
```

Same customer sees different prices based on which website they're browsing.

Multi-Currency Handling

Example: Product with different currency pricing

```
Main Website (USD):
- Price: $100.00
- Website ID: 1

UK Website (GBP):
- Price: £75.00
- Website ID: 4

EU Website (EUR):
- Price: €85.00
- Website ID: 2
```

Global Price with Regional Overrides

Scenario: Default price everywhere, but cheaper in specific region

```
Global Default:
- Customer: Beta Ltd
- Product: Widget ABC
- Quantity: 1
- Price: $100.00
- Website: All Websites (0)

EU Override:
- Customer: Beta Ltd
- Product: Widget ABC
- Quantity: 1
- Price: €80.00
- Website: EU Website (2)
```

Browsing EU website: €80.00 Browsing any other website: $100.00

Admin Setup

Step 1: Create Website-Specific Price

Navigate to: **Catalog > Customer Prices > Add New**

Step 2: Fill Basic Information

```
Customer: Select customer
Product: Select product
Quantity: 1
Price: 100.00
```

Step 3: Select Website

Website Dropdown Options:

- All Websites (ID: 0)
- Main Website (ID: 1)
- EU Website (ID: 2)
- US Website (ID: 3)

Select the specific website this price should apply to.

Step 4: Save

Click **Save** to create the website-specific price.

Advanced Scenarios

Per-Website Quantity Tiers

Example: Different tier structure by region

```
US Website (website_id=3):
qty=1,   price=$100, website_id=3
qty=50,  price=$95,  website_id=3
qty=100, price=$90,  website_id=3

EU Website (website_id=2):
qty=1,   price=€85,  website_id=2
qty=25,  price=€80,  website_id=2
qty=75,  price=€75,  website_id=2
```

Same customer, same product, but different tier breakpoints and currency per region.

Time-Based Regional Campaigns

Example: Seasonal promotion only in EU

```
EU Q1 Campaign (website_id=2):
- qty=1
- price=€70
- from_date=2025-01-01
- to_date=2025-03-31
- website_id=2

US Standard (website_id=3):
- qty=1
- price=$95
- from_date=NULL
- to_date=NULL
- website_id=3
```

EU customers see promotional price Jan-Mar, US customers see standard price year-round.

Shared Customers Across Websites

Important: If a customer can access multiple websites, they can have different prices on each:

```
Customer ID: 456 (Global B2B Customer)

On US Website:
- Sees $100.00 (website_id=3 price)

On EU Website:
- Sees €85.00 (website_id=2 price)
```

```
On APAC Website:
- Sees $110.00 (website_id=4 price)
```

Database Structure

Schema

```
CREATE TABLE pricesystem_customerprice (
    id INT PRIMARY KEY AUTO_INCREMENT,
    entity_id INT NOT NULL,
    customer_id INT NOT NULL,
    qty DECIMAL(12,4) NOT NULL,
    value DECIMAL(12,4) NOT NULL,
    from_date DATE NULL,
    to_date DATE NULL,
    website_id INT NOT NULL DEFAULT 0,
    CONSTRAINT fk_website FOREIGN KEY (website_id)
        REFERENCES store_website(website_id)
        ON DELETE CASCADE,
    CONSTRAINT UNIQUE (entity_id, customer_id, qty, from_date, to_date,
website_id)
);
```

Unique Constraint

You can have:

- Same customer/product/qty with different website_id
- Same customer/product/website with different qty
- Exact duplicate including website_id (avoid)

Foreign Key Constraint

The foreign key ensures:

- You can only use valid website IDs
- If a website is deleted, associated prices are automatically removed
- Data integrity is maintained

Price Resolution Logic

Resolution Order

When a customer views a product, the system:

1. **Find all customer prices** for entity_id + customer_id
2. **Filter by website**:
    - Include prices where website_id = current website
    - Include prices where website_id = 0 (global)
3. **Filter by date** (if from_date/to_date set)

4. **Filter by quantity** (qty <= cart quantity)
5. **Select best price**:
    - Website-specific price (if exists)
    - Global price (website_id=0) as fallback
    - Highest qty tier that matches

Example Resolution

Setup:

```
Customer: 456
Product: 123
Current Website: 2 (EU)
Cart Quantity: 15

Price Entries:
Entry A: qty=1,  price=$100, website_id=0 (global)
Entry B: qty=10, price=$95,  website_id=0 (global)
Entry C: qty=1,  price=€85,  website_id=2 (EU)
Entry D: qty=10, price=€80,  website_id=2 (EU)
```

Resolution:

1. Filter by website: Keep Entry A, B, C, D (all match website 2 or 0)
2. Filter by quantity: Keep Entry A, C, D (qty 10 ≤ 15)
3. Select best price:
    - Website-specific: Entry D (qty=10, €80, website_id=2)
    - **Entry D wins** (website-specific + highest matching qty)

Result: €80.00 applied

Frontend Display

Product Page

Shows website-specific pricing:

```
Product: Widget ABC
Your Price: €85.00

(Viewing EU Website)
```

Cart

Displays active website price:

```
Widget ABC
Website: EU Website
Quantity: 1
Unit Price: €85.00

```

Customer Account

"Download My Prices" CSV includes website info:

```
Product,SKU,Quantity,Price,Currency,Website
Widget ABC,WGT-ABC,1,85.00,EUR,EU Website
Widget ABC,WGT-ABC,1,100.00,USD,US Website
Widget ABC,WGT-ABC,1,95.00,USD,All Websites
```

Best Practices

Website Configuration

Do:

- Use website_id=0 for default/fallback pricing
- Set website-specific prices only when needed
- Document which websites use which currencies
- Test customer experience on each website
- Consider tax implications per region

Don't:

- Create prices for non-existent website IDs
- Forget to set global fallback prices
- Mix currencies on the same website
- Create unnecessary website-specific entries

Regional Strategy

Centralized Pricing

```
All pricing at website_id=0 (global)
```

Good for: Single-currency stores, simple setups

Hybrid Approach

```
Global defaults + regional overrides
```

Good for: Most multi-website scenarios

Fully Separated

```
Every website has its own price entries
```

Good for: Complex multi-currency, independent regions

Troubleshooting

Wrong Price on Website

Check:

1. Current website ID matches price entry
2. Global fallback (website_id=0) exists
3. Customer is logged in
4. Website filter in admin grid
5. Cache cleared after changes

Debug Query:

```
SELECT * FROM pricesystem_customerprice
WHERE entity_id = 123
  AND customer_id = 456
  AND (website_id = 2 OR website_id = 0)
ORDER BY website_id DESC, qty DESC;
```

Price Not Found on Website

Verify:

1. Price entry exists with correct website_id
2. Foreign key constraint hasn't deleted it
3. Website ID is valid in store_website table
4. Customer has access to that website

Duplicate Entry Error

Error Message:

```
Duplicate entry '123-456-1-NULL-NULL-2' for key 'UNIQUE'
```

Cause: Trying to create duplicate entry for same:

- entity_id + customer_id + qty + dates + website_id

Solution:

- Edit existing entry, or
- Change website_id, or
- Change qty, or
- Change date range

Combining Features

Full Feature Combination Example

```
Multi-Website + Quantity Tiers + Time-Based:
```

```
US Website Q1 Campaign:
qty=1,   price=$90,  from=2025-01-01, to=2025-03-31, website_id=3
qty=50,  price=$85,  from=2025-01-01, to=2025-03-31, website_id=3
qty=100, price=$80,  from=2025-01-01, to=2025-03-31, website_id=3

EU Website Q1 Campaign:
qty=1,   price=€80,  from=2025-01-01, to=2025-03-31, website_id=2
qty=25,  price=€75,  from=2025-01-01, to=2025-03-31, website_id=2
qty=75,  price=€70,  from=2025-01-01, to=2025-03-31, website_id=2

Global Fallback (Year-Round):
qty=1,   price=$100, from=NULL, to=NULL, website_id=0
qty=50,  price=$95,  from=NULL, to=NULL, website_id=0
```

Reporting

Prices by Website Query

```
SELECT
    w.name as website,
    c.email as customer,
    p.sku,
    cp.qty,
    cp.value
FROM pricesystem_customerprice cp
JOIN store_website w ON cp.website_id = w.website_id
JOIN customer_entity c ON cp.customer_id = c.entity_id
JOIN catalog_product_entity p ON cp.entity_id = p.entity_id
WHERE cp.website_id = 2
ORDER BY c.email, p.sku;
```

Global vs Website-Specific Count

```
SELECT
    CASE WHEN website_id = 0 THEN 'Global' ELSE 'Website-Specific' END as
type,
    COUNT(*) as count
FROM pricesystem_customerprice
GROUP BY website_id = 0;
```

Coverage Analysis

```
-- Products with website-specific pricing
SELECT
    p.sku,
    GROUP_CONCAT(DISTINCT w.name) as websites
FROM pricesystem_customerprice cp
JOIN catalog_product_entity p ON cp.entity_id = p.entity_id
JOIN store_website w ON cp.website_id = w.website_id
```

```
WHERE cp.website_id > 0
GROUP BY p.sku;
```

Migration Scenarios

Converting Global to Website-Specific

Step 1: Export existing global prices

```
SELECT * FROM pricesystem_customerprice WHERE website_id = 0;
```

Step 2: Create website-specific copies

```
INSERT INTO pricesystem_customerprice
    (entity_id, customer_id, qty, value, from_date, to_date, website_id)
SELECT
    entity_id, customer_id, qty, value, from_date, to_date, 2 as website_id
FROM pricesystem_customerprice
WHERE website_id = 0;
```

Step 3: Adjust website-specific values as needed

Step 4: Optionally remove global entries

Next Steps

- Quantity-Based Pricing Tiers
- Time-Based Pricing
- Bulk Import with Add-On

Quantity-Based Pricing Tiers

Customer Specific Prices · /customerprice/features/quantity-tiers

Overview

Quantity-based pricing allows you to offer different prices based on the quantity a customer orders. Create graduated pricing where customers receive better rates when ordering larger quantities.

How It Works

The system checks cart quantity against all price entries for the customer-product combination and automatically selects the best applicable price.

Price Selection Logic

1. Find all prices for current customer + product
2. Filter by current website (if website_id set)
3. Filter by current date (if from_date/to_date set)
4. Find entries where qty <= cart quantity
5. Select entry with highest qty that's still <= cart quantity

Creating Quantity Tiers

Example: Volume Discount

Create multiple entries with increasing quantity thresholds:

```
Customer: ACME Corp
Product: Widget ABC (SKU: WGT-ABC)

Tier 1: qty=1,   price=$100.00  (1-9 units)
Tier 2: qty=10,  price=$95.00   (10-49 units)
Tier 3: qty=50,  price=$90.00   (50-99 units)
Tier 4: qty=100, price=$85.00   (100+ units)
```

Price Application

Cart Quantity	Price Applied	Entry Used
1-9 units	$100.00	Tier 1 (qty=1)
10-49 units	$95.00	Tier 2 (qty=10)
50-99 units	$90.00	Tier 3 (qty=50)
100+ units	$85.00	Tier 4 (qty=100)

Admin Setup

Step 1: Create First Tier

Navigate to: **Catalog > Customer Prices > Add New**

```
Customer: Select customer
Product: Select product
Quantity: 1
Price: 100.00
```

Save

Step 2: Add Additional Tiers

Click "Add New" again:

```
Customer: Same customer
Product: Same product
Quantity: 10
Price: 95.00
```

Save

Repeat for each tier.

Best Practices

Tier Design

Do:

- Use standard breakpoints (1, 10, 25, 50, 100, 250, 500)
- Ensure meaningful price differences between tiers
- Document tier structure in customer contracts
- Test price display in cart

Don't:

- Create too many tiers (3-5 is optimal)
- Use odd breakpoints (e.g., qty=7, qty=23)
- Create overlapping date ranges
- Set tier prices higher than previous tier

Common Patterns

Simple Bulk Discount

```
qty=1:   $100 (base price)
qty=25:  $90  (10% off for 25+)
qty=100: $80  (20% off for 100+)
```

Fine-Grained Tiers

```
qty=1:   $10.00
qty=10:  $9.50
qty=25:  $9.00
qty=50:  $8.50
qty=100: $8.00
```

Large Volume Only

```
qty=500:  $7.50 (only activates at 500+)
qty=1000: $7.00 (1000+ units)
```

Frontend Display

Product Page

Shows available tier pricing:

- "Buy 10 for $95.00 each"
- "Buy 50 for $90.00 each"

Cart

Displays applied tier:

- Quantity: 15
- Unit Price: $95.00 (Tier 2 applied)

Customer Account

"Download My Prices" includes all tiers in CSV export.

Technical Details

Database Structure

Each tier is stored as a separate row:

id	entity_id	customer_id	qty	value	from_date	to_date	website_id
1	123	456	1	100.00	NULL	NULL	0
2	123	456	10	95.00	NULL	NULL	0
3	123	456	50	90.00	NULL	NULL	0

Unique Constraint

The combination must be unique:

- entity_id + customer_id + qty + from_date + to_date + website_id

This prevents duplicate tier definitions.

Combining with Other Features

Time-Based Tiers

Create seasonal tier pricing:

```
Q1 2025 Tiers:
qty=1,  price=$95,  from_date=2025-01-01, to_date=2025-03-31
qty=50, price=$85,  from_date=2025-01-01, to_date=2025-03-31

Q2 2025 Tiers (different pricing):
qty=1,  price=$100, from_date=2025-04-01, to_date=2025-06-30
qty=50, price=$90,  from_date=2025-04-01, to_date=2025-06-30
```

Multi-Website Tiers

Different tiers per region:

```
US Website (website_id=1):
qty=1,  price=$100
qty=50, price=$90

EU Website (website_id=2):
qty=1,  price=€85
qty=50, price=€75
```

Troubleshooting

Price Not Applying

Check:

1. Customer is logged in
2. qty value is correct (must be >= 1)
3. No date restrictions preventing activation
4. Website ID matches current website (or is 0)
5. Cache cleared after creating tiers

Wrong Tier Applied

Verify:

1. Cart quantity meets tier threshold
2. No higher tier exists with lower price
3. Date ranges don't conflict
4. Website filter is correct

Duplicate Entry Error

Cause: Attempting to create duplicate tier

Solution: Modify existing tier or change one of:

- qty value
- from_date
- to_date
- website_id

Next Steps

- Time-Based Pricing
- Multi-Website Configuration
- Bulk Import with Add-On

Time-Based Pricing

Customer Specific Prices · /customerprice/features/time-based-pricing

Overview

Configure temporary customer prices with precise date ranges. Schedule promotional prices, seasonal adjustments, or contract renewals in advance that automatically activate and deactivate.

How It Works

Time-based pricing uses two optional date fields:

- **from_date**: Price becomes active on this date (inclusive)
- **to_date**: Price expires after this date (inclusive)

The system checks these dates during price resolution and only applies prices where the current date falls within the specified range.

Date Validation Logic

```
Current Date: 2025-02-15

Price Entry 1: from_date=NULL, to_date=NULL
→ Always active

Price Entry 2: from_date=2025-01-01, to_date=NULL
→ Active (started Jan 1, no end date)

Price Entry 3: from_date=2025-03-01, to_date=2025-03-31
→ Not yet active (starts March 1)

Price Entry 4: from_date=2025-01-01, to_date=2025-01-31
→ Expired (ended Jan 31)

Price Entry 5: from_date=2025-02-01, to_date=2025-02-28
→ Active (within range)
```

Use Cases

Promotional Campaign

Scenario: Q1 2025 promotion with special pricing

```
Customer: ACME Corp
Product: Widget ABC (SKU: WGT-ABC)
Quantity: 1
Price: 85.00
From Date: 2025-01-01
```

```
To Date: 2025-03-31
```

- Before Jan 1: Standard price applies
- Jan 1 - Mar 31: $85.00 promotional price applies
- After Mar 31: Returns to standard price

Seasonal Pricing

Example: Holiday season pricing

```
Winter Season 2025:
- Customer: Beta Ltd
- Product: Widget ABC
- Quantity: 1
- Price: 95.00
- From Date: 2025-12-01
- To Date: 2026-02-28

Spring Season 2026:
- Customer: Beta Ltd
- Product: Widget ABC
- Quantity: 1
- Price: 90.00
- From Date: 2026-03-01
- To Date: 2026-05-31
```

Contract Renewal

Scenario: Transitioning from old to new contract pricing

```
Old Contract (expiring):
- Price: 100.00
- From Date: 2024-01-01
- To Date: 2024-12-31

New Contract (starting):
- Price: 95.00
- From Date: 2025-01-01
- To Date: 2025-12-31
```

Admin Setup

Step 1: Navigate to Customer Prices

Go to: **Catalog > Customer Prices > Add New**

Step 2: Fill Basic Fields

```
Customer: Select customer
Product: Select product
```

```
Quantity: 1
Price: 85.00
```

Step 3: Set Date Range

From Date:

- Click calendar icon
- Select start date: 2025-01-01
- Format: YYYY-MM-DD

To Date:

- Click calendar icon
- Select end date: 2025-03-31
- Format: YYYY-MM-DD

Step 4: Save

Click **Save** to schedule the price.

Advanced Scenarios

Multiple Time-Based Tiers

Combine time-based pricing with quantity tiers:

```
Q1 2025 Promotional Tiers:
Entry 1: qty=1,  price=$90, from=2025-01-01, to=2025-03-31
Entry 2: qty=10, price=$85, from=2025-01-01, to=2025-03-31
Entry 3: qty=50, price=$80, from=2025-01-01, to=2025-03-31

Q2 2025 Standard Tiers:
Entry 4: qty=1,  price=$95, from=2025-04-01, to=2025-06-30
Entry 5: qty=10, price=$90, from=2025-04-01, to=2025-06-30
Entry 6: qty=50, price=$85, from=2025-04-01, to=2025-06-30
```

Permanent Price with Temporary Override

```
Permanent Price:
- qty=1, price=$100, from_date=NULL, to_date=NULL

Summer Sale Override:
- qty=1, price=$80, from_date=2025-07-01, to_date=2025-08-31
```

During July and August, the $80 price applies. Outside that period, $100 applies.

Overlapping Dates

Avoid overlapping date ranges for the same customer + product + quantity combination.

Bad Example:

```
Entry 1: qty=1, price=$90, from=2025-01-01, to=2025-06-30
Entry 2: qty=1, price=$85, from=2025-03-01, to=2025-09-30
```

Problem: March-June overlap creates ambiguity.

Good Example:

```
Entry 1: qty=1, price=$90, from=2025-01-01, to=2025-02-28
Entry 2: qty=1, price=$85, from=2025-03-01, to=2025-09-30
```

Best Practices

Planning

Do:

- Schedule prices in advance
- Use full date ranges (start and end)
- Document campaigns in customer contracts
- Set reminders for upcoming price changes
- Verify dates in customer timezone

Don't:

- Create overlapping date ranges
- Use ambiguous dates (e.g., to_date without from_date)
- Forget to set end dates for temporary campaigns
- Schedule dates in the past for new entries

Date Management

Open-Ended Pricing

```
from_date=2025-01-01, to_date=NULL
```

Starts on date, never expires. Good for permanent new pricing.

Campaign Pricing

```
from_date=2025-01-01, to_date=2025-03-31
```

Fixed duration. Good for promotions and seasonal pricing.

Historical Pricing

```
from_date=2024-01-01, to_date=2024-12-31
```

Past dates. Keep for audit trail and reporting.

Database Structure

Schema

```
CREATE TABLE pricesystem_customerprice (
    id INT PRIMARY KEY AUTO_INCREMENT,
    entity_id INT NOT NULL,
    customer_id INT NOT NULL,
    qty DECIMAL(12,4) NOT NULL,
    value DECIMAL(12,4) NOT NULL,
    from_date DATE NULL,
    to_date DATE NULL,
    website_id INT NOT NULL DEFAULT 0,
    CONSTRAINT UNIQUE (entity_id, customer_id, qty, from_date, to_date,
website_id)
);
```

Unique Constraint

The combination must be unique:

- entity_id + customer_id + qty + from_date + to_date + website_id

This means you can have:

- Same product/customer/qty with different dates
- Same product/customer/dates with different qty
- Exact duplicate of all six fields

Frontend Display

Product Page

Shows available time-based pricing:

```
Current price: $100.00

Special Offers:
- From Jan 1, 2025: $85.00
- Expires: Mar 31, 2025
```

Cart

Active time-based price is applied:

```
Widget ABC
Quantity: 1
Unit Price: $85.00 (Q1 2025 Promotion)
```

Customer Account

"Download My Prices" CSV includes date ranges:

```
Product,SKU,Quantity,Price,From Date,To Date
Widget ABC,WGT-ABC,1,85.00,2025-01-01,2025-03-31
Widget ABC,WGT-ABC,1,100.00,,
```

Troubleshooting

Price Not Activating

Check:

1. Current date is >= from_date
2. Current date is <= to_date (if set)
3. Date format is correct (YYYY-MM-DD)
4. Server timezone matches expected timezone
5. Cache has been cleared

Debug Query:

```
SELECT * FROM pricesystem_customerprice
WHERE entity_id = 123
  AND customer_id = 456
  AND (from_date IS NULL OR from_date <= CURDATE())
  AND (to_date IS NULL OR to_date >= CURDATE());
```

Price Not Expiring

Verify:

1. to_date is set correctly
2. to_date is in the past
3. No other active price entry exists
4. Cache has been cleared

Wrong Price Applied

Common Causes:

1. Multiple entries with overlapping dates
2. Quantity tier overriding time-based price
3. Website filter excluding current store
4. Cache showing old price

Solution: Review all entries for customer + product:

```
SELECT qty, value, from_date, to_date, website_id
FROM pricesystem_customerprice
WHERE entity_id = 123 AND customer_id = 456
```

```
ORDER BY from_date DESC, qty ASC;
```

Combining Features

Time-Based + Quantity Tiers + Multi-Website

```
US Website (website_id=1), Q1 2025:
qty=1,  price=$90, from=2025-01-01, to=2025-03-31, website_id=1
qty=50, price=$85, from=2025-01-01, to=2025-03-31, website_id=1

EU Website (website_id=2), Q1 2025:
qty=1,  price=€80, from=2025-01-01, to=2025-03-31, website_id=2
qty=50, price=€75, from=2025-01-01, to=2025-03-31, website_id=2
```

Reporting

Active Campaigns Query

```
SELECT
    c.email as customer,
    p.sku,
    cp.qty,
    cp.value,
    cp.from_date,
    cp.to_date
FROM pricesystem_customerprice cp
JOIN customer_entity c ON cp.customer_id = c.entity_id
JOIN catalog_product_entity p ON cp.entity_id = p.entity_id
WHERE cp.from_date <= CURDATE()
  AND (cp.to_date IS NULL OR cp.to_date >= CURDATE())
ORDER BY cp.to_date ASC;
```

Upcoming Campaigns Query

```
SELECT * FROM pricesystem_customerprice
WHERE from_date > CURDATE()
ORDER BY from_date ASC;
```

Expired Campaigns Query

```
SELECT * FROM pricesystem_customerprice
WHERE to_date < CURDATE()
ORDER BY to_date DESC
LIMIT 100;
```

Next Steps

- Quantity-Based Pricing Tiers
- Multi-Website Configuration
- Bulk Import with Add-On

Advanced Config

Customer Specific Prices · /customerprice/addons/advanced-config

The Advanced Config add-on extends Pricesystem with additional configuration and override options that affect how **customer prices** are selected and applied.

Package

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

Related

- Price Selection
- Pricesystem Advanced Config

CSV Import/Export

Customer Specific Prices · /customerprice/addons/import-export

Bulk import and export **customer prices** via CSV.

Package

- Composer package: `mageb2b/pricesystem-customerprice-importexport`
- Magento module: `MageB2B_PricesystemCustomerpriceImportExport`

Installation

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

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

Admin Import/Export (Magento ImportExport)

Use Magento's importer:

1. Go to **System > Data Transfer > Import**
2. Entity type: **Pricesystem Customerprice**

Tip: Export first and use the exported CSV as your template (column names vary by version/config).

CSV Columns (Customerprice)

Common columns:

- `customer_id` (required): customer ID **or** customer email
- `product_id` (required): product ID **or** SKU (depends on config `pricesystem/customerprice_import/product_id_alias`)
- `price` (required): numeric value
- `qty` (optional): quantity tier
- `website_id` (optional): website scope
- `from_date`, `to_date` (optional): date range
- `price_application_type` (optional): `fixed`, `surcharge_nominal`, `surcharge_percent`, `discount_nominal`, `discount_percent`
- `price_attribute_id` (optional): custom price attribute mapping (if used in your setup)

CLI Commands

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

Useful flags (example):

```
php bin/magento pricesystem:import-customerprice /path/to/customerprice.csv
--behavior=add_update
php bin/magento pricesystem:export-customerprice --
export_path=var/export/customerprice.csv
```

Related

- My Prices
- Price Selection

SOAP / REST API

Customer Specific Prices · /customerprice/addons/rest-api

Adds CRUD WebAPI endpoints for **customer prices**.

Package

- Composer package: `mageb2b/pricesystem-customerprice-api`
- Magento module: `MageB2B_PricesystemCustomerpriceApi`

Installation

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

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

Endpoints

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`

Related

- Pricesystem Core API

Sample Data

Customer Specific Prices · /customerprice/addons/sample-data

Installs demo entities and example **customer price** records.

Packages

- Core sample data: `mageb2b/pricesystem-sample-data` (`MageB2B_PricesystemCoreSampleData`)
- Customerprice sample data: `mageb2b/pricesystem-customerprice-sample-data` (`MageB2B_PricesystemCustomerpriceSampleData`)

Installation

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

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

Related

- Pricesystem Sample Data

Price Selection

Customer Specific Prices · /customerprice/price-selection

Customerprice is a Pricesystem price type. Pricesystem calculates multiple candidate price types and then selects the final purchasable price using a configurable strategy.

This section documents the global selection logic, because it affects how `customer_price` competes with other price types (special price, tier price, pricelist, etc.).

Quick Links

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

Customerprice In The Strategy

- Price code: `customer_price`
- This price type is only available if the Customerprice module is installed and a matching customer price exists for the current context.

Tip: if you want customer-specific prices to win, put `customer_price` near the top of your sort order list.

Customer Group Overrides

Customer Specific Prices · /customerprice/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

When Group Overrides Apply

- For **guests**: group-level selection applies (NOT_LOGGED_IN group).
- For **logged-in customers**: group-level selection applies only if the customer-level rule is set to **Fallback**.

Group vs System Configuration

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

- `pricesystem/price_select_rule/priceselect`

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, 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 + Fallback

Customer Specific Prices · /customerprice/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.

System Config

If set to "System Config", Pricesystem uses the global strategy from:

- `pricesystem/price_select_rule/priceselect`

and it does **not** fall back to the group rule.

Fallback

If set to "Fallback", Pricesystem tries:

1. Group-level price select rule (if configured)
2. Otherwise, the global system configuration

This is the recommended mode if you want customer group rules to apply consistently (including for customers that also have `customer_price` entries).

Sort order (from config)

Uses the global sort order list from system configuration.

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.

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

Recommended Default For New Customers

Advanced Config provides a default for new customers:

- Config path: `pricesystem/advanced/default_priceselect`

Recommendation for Customerprice:

- Set the default to **Fallback** so group-level pricing rules still apply by default.

Formula Pricing

Customer Specific Prices · /customerprice/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")

Variables (Price Codes)

To reference Customerprice in formulas, use:

- `customer_price`

You can also reference other price codes such as `special_price`, `tier_price`, `catalog_rule_price`, `orig_price`, etc.

Notes:

- Both `code_with_underscores` and `codewithunderscores` are accepted.
- `orig_price` is always available as a reference for fallback and for formulas.

Missing Inputs And Fallback

Formula behavior is controlled by system configuration:

- Missing values behavior: `pricesystem/price_select_rule/formula_missing_input_behavior`
- Formula error fallback: `pricesystem/price_select_rule/formula_fallback_strategy`

Sort Order Strategy

Customer Specific Prices · /customerprice/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.

This directly controls how `customer_price` competes with other price types (special price, tier price, pricelist, etc.).

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).

Recommended Priority For Customerprice Stores

If you want customer-specific pricing to win, a common priority is:

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

Exact ordering depends on your installed price types and business rules.

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. `customer_price`
2. `special_price`
3. `orig_price`

If `customer_price` is selected as the final price, the strike-through price can become `special_price` (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)

Customer Specific Prices · /customerprice/price-selection/system-configuration

Even if you only purchased Customerprice, the Pricesystem core module is always installed, and the global selection rules are configured there:

- 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 (and `customer_price` will not be selected).

Strategy: Price Select Rule

- 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)

If you want customer-specific prices to win, put `customer_price` near the top of the list.

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 price can be the next available price from your sort order list.

Related:

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

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.