Category Specific Prices

PDF Manual

MODULE

categoryprice

LAST UPDATED

2026-03-29

Priority System

Category Specific Prices · /categoryprice/features/priority-system

Smart conflict resolution when multiple category prices match the same customer and category.

Overview

The Priority System determines which price wins when conflicts occur. Conflicts happen when:

- Same customer has multiple prices for same category
- Customer has both individual price AND group price
- Multiple quantity tiers match
- Multiple date ranges overlap

The system resolves these conflicts predictably and transparently.

How Priority Works

Priority Field

Location: Every price record has a `priority` field

Type: Integer (0-999)

Default: 0

Rule: Higher priority wins

Example:

- Priority 10 vs Priority 20 → **Priority 20 wins**
- Priority 5 vs Priority 50 → **Priority 50 wins**
- Priority 0 vs Priority 0 → **First match wins** (undefined behavior)

Configuration

Global Price Select Rule

Location: Stores > Configuration > Pricesystem > Categoryprice Settings

Setting: "Price Select Rule"

Options:

1. Select price by priority (Default)

- Uses numeric priority field to determine winner

- Higher priority value takes precedence
- Most flexible and powerful option
- Recommended for complex pricing scenarios

2. Group price first

- Always prefers customer group prices over individual customer prices
- Ignores priority field for customer vs group conflicts
- Still uses priority for group vs group or customer vs customer
- Simpler logic, less flexible

3. Customer price first

- Always prefers individual customer prices over group prices
- Ignores priority field for customer vs group conflicts
- Still uses priority for group vs group or customer vs customer
- Good for customer-centric pricing strategies

Conflict Scenarios

Scenario 1: Customer vs Customer (Same Customer)

Setup:

```
Price 1: Customer #123 + Electronics + $100 + Priority 10
Price 2: Customer #123 + Electronics + $90 + Priority 20
```

Resolution:

- **Both match:** Same customer, same category
- **Winner:** Price 2 ($90) - Priority 20 > 10
- **Configuration:** Doesn't matter (same customer type)

Use Case: Promotional override of standard pricing.

Scenario 2: Group vs Group (Impossible)

Setup:

```
Price 1: Wholesale Group + Electronics + $90 + Priority 15
Price 2: Retail Group + Electronics + $100 + Priority 10
```

Resolution:

- **Conflict:** Never occurs!
- **Why:** Customers belong to exactly ONE group
- **Result:** Only the customer's group price matches

Note: Magento doesn't support customers in multiple groups simultaneously.

Scenario 3: Customer vs Group

Setup:

```
Customer Price: Customer #123 + Electronics + $95 + Priority 15
Group Price: Wholesale + Electronics + $85 + Priority 25
Customer #123 is in Wholesale group
```

Resolution depends on configuration:

With "Select price by priority"

- **Both match:** Customer #123 matches both prices
- **Winner:** Group price ($85) - Priority 25 > 15
- **Logic:** Priority field determines winner

With "Customer price first"

- **Both match:** Customer #123 matches both prices
- **Winner:** Customer price ($95)
- **Logic:** Customer always beats group, priority ignored

With "Group price first"

- **Both match:** Customer #123 matches both prices
- **Winner:** Group price ($85)
- **Logic:** Group always beats customer, priority ignored

Scenario 4: Quantity Tier Conflicts

Setup:

```
Price 1: Customer #123 + Electronics + Qty 1 + $100 + Priority 10
Price 2: Customer #123 + Electronics + Qty 1 + $95 + Priority 20
```

Cart:

- Customer #123 ordering 5 items from Electronics

Resolution:

- **Both match:** Qty 1 ≤ 5 for both prices
- **Winner:** Price 2 ($95) - Priority 20 > 10
- **Result:** All 5 items at $95

Note: Quantity tiers are resolved BEFORE priority. The system first finds all matching qty tiers, then applies priority among them.

Scenario 5: Date Range Conflicts

Setup:

```
Price 1: Wholesale + Electronics + $90 + Priority 10
  From: 2025-01-01, To: 2025-12-31

Price 2: Wholesale + Electronics + $85 + Priority 25
  From: 2025-06-01, To: 2025-08-31 (Summer sale)
```

Timeline:

January - May:

- Only Price 1 is active
- Result: $90

June - August:

- BOTH prices active (dates overlap)
- Conflict!
- Winner: Price 2 ($85) - Priority 25 > 10
- Result: $85 (summer sale)

September - December:

- Only Price 1 is active again
- Result: $90

Key Point: Higher priority summer sale overrides standard pricing during summer months.

Priority Value Recommendations

Standard Priority Hierarchy

0-9: System Defaults

- Priority 0: Fallback pricing
- Priority 5: Base prices

10-19: Standard Pricing

- Priority 10: Regular customer prices
- Priority 12: Regular wholesale prices
- Priority 15: Regular VIP prices

20-29: Promotional Pricing

- Priority 20: Standard promotions

- Priority 22: Seasonal campaigns
- Priority 25: Flash sales

30-39: VIP/Contract Overrides

- Priority 30: VIP contract pricing
- Priority 35: Special agreements

40-49: Emergency Overrides

- Priority 40: Temporary price fixes
- Priority 45: Urgent promotions

50+: Reserved

- Priority 50+: System overrides
- Priority 99: Testing
- Priority 999: Absolute override

Resolution Algorithm

Step-by-Step Process

When a customer views a product in a category, the system executes:

```
1. Get all category prices (both customer and group tables)

2. Filter by customer match:
   - Customer-specific prices for this customer ID
   - Group prices for this customer's group ID

3. Filter by category match:
   - entity_id matches category ID of product

4. Filter by website match:
   - website_id matches current website
   - OR website_id = 0 (all websites)

5. Filter by date validity:
   - (from_date IS NULL OR from_date <= today)
   - AND (to_date IS NULL OR to_date >= today)

6. Filter by quantity:
   - price.qty <= cart_quantity
   - Select highest matching qty tier

7. Apply priority resolution:
   a. If "Select by priority": Sort by priority DESC
   b. If "Customer first": Customer prices, then groups
   c. If "Group first": Group prices, then customers
```

```
8. Select top result:
   - First price after sorting
   - Apply to product
```

Configuration Examples

Example 1: Customer-Centric Strategy

Business Rule: "Individual customer agreements always take precedence."

Configuration:

- Price Select Rule: "Customer price first"

Behavior:

- Customer price: $95, Priority 10
- Group price: $85, Priority 30
- **Winner:** Customer price ($95)
- **Reason:** Customer always wins, priority ignored

Best For:

- Contract-based businesses
- Custom negotiated pricing
- Account management focus

Example 2: Volume-Focused Strategy

Business Rule: "Wholesale groups get best prices, individuals get standard pricing."

Configuration:

- Price Select Rule: "Group price first"

Behavior:

- Customer price: $100, Priority 30
- Group price: $85, Priority 10
- **Winner:** Group price ($85)
- **Reason:** Group always wins, priority ignored

Best For:

- Volume discount programs
- Wholesale-first businesses
- Group pricing incentives

Example 3: Flexible Strategy (Recommended)

Business Rule: "Use priority to handle specific cases dynamically."

Configuration:

- Price Select Rule: "Select price by priority"

Behavior:

```
Standard customer: $95, Priority 10
Wholesale group: $90, Priority 15
VIP customer: $85, Priority 30
Summer sale group: $80, Priority 25
```

Results:

- VIP customer wins: $85 (Priority 30)
- During summer: VIP still wins (30 > 25)
- Non-VIP in summer: Summer sale wins (25 > 15)

Best For:

- Complex pricing strategies
- Multiple overlapping campaigns
- Fine-grained control

Priority Management

Setting Priorities

When Creating Prices:

1. Consider the price purpose
2. Check existing priorities for similar prices
3. Set new priority accordingly
4. Document priority reasoning

Example Workflow:

1. Check: "What wholesale prices exist?"
2. Find: Priority 12 for standard wholesale
3. New seasonal wholesale: Priority 22 (promotional tier)
4. Document: "Summer wholesale sale, overrides standard wholesale"

Auditing Priorities

Quarterly Review:

```
-- Find conflicting priorities (same customer+category+qty)
SELECT entity_id, customer_id, qty, COUNT(*) as conflicts
FROM pricesystem_categoryprice
GROUP BY entity_id, customer_id, qty
HAVING COUNT(*) > 1;

-- Find conflicting group priorities
SELECT entity_id, group_id, qty, COUNT(*) as conflicts
FROM pricesystem_categoryprice_customergroup
GROUP BY entity_id, group_id, qty
HAVING COUNT(*) > 1;
```

Resolution:

1. Review business reason for conflicts
2. Adjust priorities to match intent
3. Document exceptions

Bulk Priority Updates

Scenario: Increase all VIP priorities by 10.

```
-- Customer prices
UPDATE pricesystem_categoryprice
SET priority = priority + 10
WHERE customer_id IN (
  SELECT entity_id FROM customer_entity WHERE group_id = 4
);

-- Group prices
UPDATE pricesystem_categoryprice_customergroup
SET priority = priority + 10
WHERE group_id = 4;
```

Advanced Scenarios

Scenario A: Three-Tier Override

Setup:

```
Base:       Wholesale + All Categories + $100 + Priority 10
Campaign:   Wholesale + Electronics + $90 + Priority 20
VIP:        Customer #123 + Electronics + $85 + Priority 30
```

Customer #123 in Wholesale group browsing Electronics:

Evaluation:

1. Base wholesale: Matches ($100, P10)
2. Electronics campaign: Matches ($90, P20)
3. VIP customer: Matches ($85, P30)

Winner: VIP price ($85, Priority 30)

Logic: All three match, highest priority wins.

Scenario B: Date-Based Priority Shift

Setup:

```
Standard:   Wholesale + Electronics + $100 + Priority 15
            From: NULL, To: NULL

Black Friday: Wholesale + Electronics + $75 + Priority 30
              From: 2025-11-29, To: 2025-12-02

Cyber Monday: Wholesale + Electronics + $80 + Priority 35
              From: 2025-12-02, To: 2025-12-03
```

Timeline:

November 28:

- Only Standard active: $100

November 29-30 (Black Friday):

- Standard + Black Friday active
- Winner: Black Friday ($75, P30 > P15)

December 2 (Cyber Monday):

- Standard + Black Friday + Cyber Monday active
- Winner: Cyber Monday ($80, P35 > P30)

December 3+:

- Only Standard active: $100

Key: Temporary campaigns override standard with higher priority, expire automatically.

Scenario C: Customer Override of Group Campaign

Business Rule: "VIP customers get better pricing than group campaigns."

Setup:

```
Group Campaign: Wholesale + Electronics + $85 + Priority 25
                From: 2025-06-01, To: 2025-08-31

VIP Standard:   Customer #123 + Electronics + $80 + Priority 30
                From: NULL, To: NULL
```

Result:

- VIP customer: Always $80 (Priority 30 > 25), even during campaign
- Regular wholesale: $85 during summer campaign

Purpose: VIP customer agreements always honored, even during promotions.

Testing Priority Logic

Manual Testing

Test Case 1: Basic Priority

1. Create: Customer #TEST + Category TEST + $100 + Priority 10
2. Create: Customer #TEST + Category TEST + $90 + Priority 20
3. Log in as customer TEST
4. Browse category TEST product
5. **Expected:** $90 displays
6. **Verify:** Priority 20 won

Test Case 2: Configuration Switch

1. Config: "Select price by priority"
2. Create: Customer #TEST + Electronics + $95 + Priority 15
3. Create: Wholesale Group + Electronics + $85 + Priority 25
4. Assign customer TEST to Wholesale
5. **Expected:** $85 (priority 25)
6. Change config: "Customer price first"
7. **Expected:** $95 (customer wins, priority ignored)
8. Change config: "Group price first"
9. **Expected:** $85 (group wins, priority ignored)

Automated Testing

PHPUnit Test Example:

```
public function testPriorityResolution()
{
    $customer = $this->createCustomer();
    $category = $this->createCategory();

    // Create two prices
    $this->createCategoryPrice([
```

```
        'customer_id' => $customer->getId(),
        'entity_id' => $category->getId(),
        'value' => 100,
        'priority' => 10
    ]);

    $this->createCategoryPrice([
        'customer_id' => $customer->getId(),
        'entity_id' => $category->getId(),
        'value' => 90,
        'priority' => 20
    ]);

    $price = $this->priceResolver->resolve($customer, $category);

    $this->assertEquals(90, $price);
}
```

Troubleshooting

Wrong Price Wins

Symptom: Lower priority price is being applied.

Checklist:

1. ✓ Check configuration: "Select by priority" enabled?
2. ✓ Verify priority values in database
3. ✓ Clear cache: `php bin/magento cache:flush`
4. ✓ Check Pricesystem Core price order
5. ✓ Verify customer/group assignment
6. ✓ Check date ranges (might filter out high priority)

Debug SQL:

```
SELECT
  'customer' as type,
  customer_id,
  NULL as group_id,
  value,
  priority,
  from_date,
  to_date
FROM pricesystem_categoryprice
WHERE customer_id = 123 AND entity_id = 456

UNION ALL

SELECT
```

```
  'group' as type,
  NULL as customer_id,
  group_id,
  value,
  priority,
  from_date,
  to_date
FROM pricesystem_categoryprice_customergroup
WHERE group_id = 2 AND entity_id = 456

ORDER BY priority DESC;
```

Configuration Not Applying

Symptom: Changed config, no effect.

Steps:

1. Clear config cache: `php bin/magento cache:clean config`
2. Clear full cache: `php bin/magento cache:flush`
3. Verify config saved: Check database `core_config_data` table
4. Check config scope: Store view vs website vs global
5. Re-index if needed: `php bin/magento indexer:reindex`

Undefined Behavior

Symptom: Inconsistent pricing with same priorities.

Cause: Multiple prices with identical priority.

Example:

```
Price 1: $100, Priority 10
Price 2: $90, Priority 10
```

Problem: Both priority 10, winner undefined (might be insertion order, ID order, or random).

Solution: Assign unique priorities:

```
Price 1: $100, Priority 10
Price 2: $90, Priority 11
```

Best Practices

1. Use Priority Tiers

Don't use arbitrary values. Use consistent tiers:

- 0-9, 10-19, 20-29, 30-39, etc.

2. Document Priority Meanings

Create internal documentation:

- "Priority 15 = Standard wholesale"
- "Priority 25 = Promotional campaigns"
- "Priority 30 = VIP contracts"

3. Leave Gaps

Don't use consecutive values:

- 10, 15, 20, 25, 30
- 10, 11, 12, 13, 14

Gaps allow inserting priorities later without renumbering.

4. Test Configuration Changes

Before changing global config:

1. Test in staging environment
2. Verify expected behavior
3. Document expected changes
4. Schedule during low traffic
5. Monitor after deployment

5. Audit Regularly

Quarterly review:

- Find conflicts (same priority)
- Find obsolete prices (expired dates)
- Find unused priorities (gaps in sequence)
- Adjust as needed

Related Documentation

- Creating Category Prices
- Category Customer Pricing
- Category Group Pricing