Product-Customer Matrix Prices

PDF Manual

MODULE

productcustomermatrix

LAST UPDATED

2026-03-29

Priority-Based Resolution

Product-Customer Matrix Prices · /productcustomermatrix/features/priority-resolution

Handle scenarios where customers match multiple matrices using numeric priority values and merge strategies.

Overview

Customers can match multiple matrices through:

- Different matching attributes (group + region + company)
- Manual matrix assignments
- Overlapping date ranges
- Broad attribute rules (OR logic)

The Priority System resolves these conflicts predictably and transparently.

Priority Field

Every matrix has a `priority` field:

- **Type:** Integer
- **Range:** 0-999
- **Default:** 0
- **Rule:** Higher value = higher precedence

Example:

- Matrix A: Priority 10
- Matrix B: Priority 20
- **Winner:** Matrix B

Configuration

Merge Matrix Quantities

Path: Stores > Configuration > Pricesystem > Product-Customer Matrix > Merge Matrix Qtys

Setting: `pricesystem/productcustomermatrix/merge_matrix_qtys`

Values:

No (Default): Select Highest Priority

When customer matches multiple matrices, use ONLY the highest priority matrix.

Example:

```
Customer matches:
```

```
- Matrix A (Priority 10): Product X @ qty=1 ($100), qty=10 ($95)
- Matrix B (Priority 20): Product X @ qty=1 ($98), qty=50 ($90)

Result: Use ONLY Matrix B (higher priority)
- qty=1: $98
- qty=50: $90
(Matrix A ignored completely)
```

Use Case: Simple, predictable pricing. One matrix wins.

Yes: Merge Quantity Tiers

Combine quantity tiers from ALL matching matrices, taking best price at each tier.

Example:

```
Customer matches:
- Matrix A (Priority 10): Product X @ qty=1 ($100), qty=10 ($95)
- Matrix B (Priority 20): Product X @ qty=1 ($98), qty=50 ($90)

Result: MERGE tiers, best price wins:
- qty=1: $98 (from Matrix B, better than A's $100)
- qty=10: $95 (from Matrix A, B has no qty=10)
- qty=50: $90 (from Matrix B, A has no qty=50)

Customer gets: qty=1 ($98), qty=10 ($95), qty=50 ($90)
Best of both matrices!
```

Use Case: Complex pricing scenarios, best-of-all-matrices strategy.

Priority Hierarchy Recommendations

Standard Hierarchy

0-9: Base/Fallback

- Priority 0: Base matrix (website default)
- Priority 5: Legacy/migration pricing

10-19: Standard Pricing

- Priority 10: Standard retail
- Priority 12: Standard wholesale
- Priority 15: Regional standard

20-29: Promotional Pricing

- Priority 20: Standard promotions
- Priority 22: Seasonal sales
- Priority 25: Flash sales

30-39: VIP/Contract

- Priority 30: VIP tier pricing
- Priority 32: Annual contracts
- Priority 35: Enterprise agreements

40-49: Emergency Overrides

- Priority 40: Temporary fixes
- Priority 45: Urgent campaigns

50+: Reserved

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

Resolution Scenarios

Scenario 1: Same Priority

Setup:

```
Matrix A: Priority 10
Matrix B: Priority 10
```

Merge = No:

- **Undefined behavior!** (might be ID order, creation order, or random)
- **Solution:** Assign unique priorities (10 vs 11)

Merge = Yes:

- Merges tiers from both
- Best price wins at each tier
- Defined behavior

Best Practice: Avoid same-priority conflicts by using unique values.

Scenario 2: Clear Winner

Setup:

```
Matrix A: Priority 10 (Standard wholesale)
Matrix B: Priority 30 (VIP contract)
```

Merge = No:

- Use Matrix B only (Priority 30 > 10)

Merge = Yes:

- Merge tiers from both
- VIP contract prices likely win at each tier (better pricing)
- But if wholesale has unique tiers, customer gets those too

Scenario 3: Three-Way Conflict

Setup:

```
Customer matches three matrices:
- Matrix A: Priority 15 (Wholesale)
- Matrix B: Priority 20 (California Regional)
- Matrix C: Priority 30 (ACME Contract)

Product X prices:
- A: qty=1 ($100), qty=10 ($95)
- B: qty=1 ($98), qty=25 ($92)
- C: qty=1 ($96), qty=50 ($88)
```

Merge = No:

- Use ONLY Matrix C (Priority 30)
- Customer gets: qty=1 ($96), qty=50 ($88)

Merge = Yes:

- Merge all three matrices
- Best price at each tier:
    - qty=1: $96 (from C)
    - qty=10: $95 (from A)
    - qty=25: $92 (from B)
    - qty=50: $88 (from C)
- Customer gets: qty=1 ($96), qty=10 ($95), qty=25 ($92), qty=50 ($88)

Analysis: Merge gives best pricing across all tiers.

Scenario 4: Date-Based Priority Shift

Setup:

```
Standard pricing:
- Matrix: "Wholesale 2025"
- Priority: 15
- Dates: 2025-01-01 to 2025-12-31
- Product X: qty=1 ($100)
```

```
Campaign:
- Matrix: "Black Friday 2025"
- Priority: 25
- Dates: 2025-11-29 to 2025-12-02
- Product X: qty=1 ($75)
```

Timeline:

Nov 28:

- Only Wholesale active: $100

Nov 29-30 (Black Friday):

- Both active, campaign wins (Priority 25 > 15): $75

Dec 3+:

- Only Wholesale active again: $100

Key Point: Temporary high-priority campaigns override standard pricing.

Scenario 5: Product Not in High-Priority Matrix

Setup:

```
Matrix A (Priority 10): Products X, Y, Z
Matrix B (Priority 20): Products X, Y (Z missing)
```

Customer views Product Z:

Merge = No:

- Use Matrix B (higher priority)
- Product Z not in Matrix B
- **Result:** No matrix price (fallback to next pricesystem module)

Merge = Yes:

- Check all matrices
- Product Z found in Matrix A
- **Result:** Uses Matrix A price for Product Z

Lesson: Merge mode more forgiving when high-priority matrices have incomplete product sets.

Scenario 6: Customer-Specific Date Override

Setup:

```
Matrix: "Annual Contract - ACME" (Priority 35)
- Matrix Dates: 2025-01-01 to 2025-12-31
- Company: ACME

Customer #123:
- Automatic assignment via company match
- Manual override dates: 2025-01-01 to 2025-06-30 (6-month trial)

Customer #456:
- Automatic assignment via company match
- No override dates (uses matrix dates)
```

Result:

- Customer #123: Matrix active Jan 1 - Jun 30 only
- Customer #456: Matrix active Jan 1 - Dec 31 (full year)

Key Point: Customer-specific dates override matrix dates, allowing per-customer scheduling within same matrix.

Scenario 7: Priority + Merge with Overlapping Tiers

Setup:

```
Matrix A (Priority 10):
- Product X: qty=1 ($100), qty=10 ($90), qty=50 ($80)

Matrix B (Priority 20):
- Product X: qty=1 ($95), qty=25 ($85), qty=100 ($75)

Matrix C (Priority 30):
- Product X: qty=1 ($98), qty=50 ($78)
```

Customer orders 40 units.

Merge = No (Highest Priority Only):

```
Use Matrix C only (Priority 30)
Best tier for qty=40: qty=1 ($98)
Result: $98/unit × 40 = $3,920
```

Merge = Yes (Best Price Wins):

```
Merge all tiers:
- qty=1: Best of ($100, $95, $98) = $95 (Matrix B)
- qty=10: Best of ($90, $95, $98) = $90 (Matrix A)
- qty=25: Best of ($80, $85, $98) = $80 (Matrix A)
- qty=50: Best of ($80, $75, $78) = $75 (Matrix B)
```

```
- qty=100: $75 (Matrix B)

Customer orders 40 units → Uses qty=25 tier: $80
Result: $80/unit × 40 = $3,200

Savings: $720 by using merge!
```

Analysis: Merge can significantly benefit customers by combining best prices from multiple matrices.

Priority Management

Setting Priorities

At Creation:

1. Identify purpose (standard/promo/VIP/contract)
2. Check existing matrices in same tier
3. Assign priority within tier
4. Document reasoning

Example:

```
Creating "Summer Sale 2025"
Purpose: Seasonal promotion
Tier: 20-29 (promotional)
Existing: Spring Sale (22)
Assign: 23 (next in sequence)
Document: "Summer campaign, overrides standard wholesale (15)"
```

Changing Priorities

Caution: Affects all customers assigned to matrix.

Process:

1. Understand current behavior
2. Test priority change in staging
3. Verify customer impact
4. Update during low-traffic period
5. Monitor after change
6. Document change reason

Bulk Priority Updates

Scenario: Increase all VIP priorities by 10.

SQL:

```
UPDATE pricesystem_product_customer_matrix
```

```
SET priority = priority + 10
WHERE name LIKE 'VIP%'
OR id IN (
  SELECT matrix_id FROM pricesystem_product_customer_matrix_attribute
  WHERE attribute_code = 'group' AND attribute_value = '4' -- VIP group
);
```

Caution: Test first! Affects all customers.

Testing Priority Logic

Test Case 1: Basic Priority

Setup:

1. Create Matrix A (Priority 10) with Product X = $100
2. Create Matrix B (Priority 20) with Product X = $90
3. Assign test customer to both matrices (manually)

Test Merge = No:

- Expected: $90 (Matrix B wins)
- Verify: Log in, check Product X price

Test Merge = Yes:

- Expected: $90 (still $90, both have same tier)
- Verify: Same result

Test Case 2: Merge Benefit

Setup:

1. Matrix A (Priority 10): Product X @ qty=1 ($100), qty=10 ($95)
2. Matrix B (Priority 20): Product X @ qty=1 ($98), qty=50 ($90)
3. Assign test customer to both

Test Merge = No:

- Expected: Only qty=1 ($98), qty=50 ($90) from Matrix B
- Verify: qty=10 tier NOT available

Test Merge = Yes:

- Expected: qty=1 ($98), qty=10 ($95), qty=50 ($90)
- Verify: qty=10 tier available (from Matrix A)

Test Case 3: Date-Based Priority Shift

Setup:

1. Matrix A: Standard pricing (Priority 15, permanent)

2. Matrix B: Campaign pricing (Priority 25, 7 days only)
3. Assign customer to both

Test Before Campaign:

- Expected: Matrix A price
- Verify: Check product price

Test During Campaign:

- Expected: Matrix B price (higher priority)
- Verify: Price changed

Test After Campaign:

- Expected: Matrix A price again
- Verify: Price reverted

Best Practices

1. Use Tiers, Not Random Values

Good:

- 0, 10, 20, 30, 40 (clear tiers)
- 10, 15, 20, 25, 30 (with gaps)

Bad:

- 7, 13, 21, 29, 33 (no pattern)
- 1, 2, 3, 4, 5 (no gaps)

2. Leave Gaps

Allows inserting priorities later without renumbering.

Example:

- Current: 10, 20, 30
- Need to insert between 10 and 20: Use 15
- With consecutive (10, 11, 12): Must renumber everything

3. Document Hierarchy

Create internal documentation:

```
Priority Hierarchy - MageB2B Product-Customer Matrices

0-9: Base matrices
  0 = Website default pricing

10-19: Standard pricing
```

```
  10 = Standard retail
  12 = Standard wholesale
  15 = Regional standard

20-29: Promotional pricing
  20 = Standard promotions
  25 = Flash sales/urgent campaigns

30-39: VIP/Contract pricing
  30 = VIP tier
  35 = Enterprise contracts

40+: Emergency/testing
```

4. Test Merge Configuration

Before enabling merge:

1. Test with 2-3 matrices
2. Verify expected behavior
3. Check edge cases (missing products, same tiers)
4. Monitor after enabling

5. Avoid Priority 999

Reserve for absolute emergencies only. Creates maintenance issues.

6. Audit Regularly

Quarterly review:

- List matrices by priority
- Find duplicates (same priority)
- Find gaps (orphaned tiers)
- Verify hierarchy still makes sense

Troubleshooting

Wrong Price Displaying

Checklist:

1. ✓ Which matrices does customer match?
2. ✓ What are their priorities?
3. ✓ Is merge enabled?
4. ✓ Does product exist in high-priority matrix?
5. ✓ Are dates valid for all matrices?
6. ✓ Cache cleared?

Debug SQL:

```
-- Find customer's matrices
SELECT
  m.id,
  m.name,
  m.priority,
  m.is_active,
  m.from_date,
  m.to_date,
  CASE
    WHEN mc.from_date IS NOT NULL THEN mc.from_date
    ELSE m.from_date
  END AS effective_from_date,
  CASE
    WHEN mc.to_date IS NOT NULL THEN mc.to_date
    ELSE m.to_date
  END AS effective_to_date
FROM pricesystem_product_customer_matrix m
JOIN pricesystem_product_customer_matrix_customer mc ON mc.matrix_id = m.id
WHERE mc.customer_id = 123
  AND m.is_active = 1
ORDER BY m.priority DESC;

-- Check product prices in those matrices
SELECT
  m.name,
  m.priority,
  p.qty,
  p.price
FROM pricesystem_pricelist_product p
JOIN pricesystem_product_customer_matrix m ON m.id = p.pricelist_id
WHERE p.product_id = 456
  AND p.pricelist_id IN (
    SELECT matrix_id FROM pricesystem_product_customer_matrix_customer
    WHERE customer_id = 123
  )
ORDER BY m.priority DESC, p.qty ASC;
```

Merge Not Working

Symptoms: Customer only sees one matrix's prices, not merged.

Checks:

1. ✓ `merge_matrix_qtys = Yes` in config?
2. ✓ Config cache cleared?
3. ✓ Customer actually assigned to multiple matrices?
4. ✓ Matrices have different qty tiers to merge?

Test SQL:

```
-- Check merge config
SELECT * FROM core_config_data
```

```
WHERE path = 'pricesystem/productcustomermatrix/merge_matrix_qtys';

-- Check customer's matrices count
SELECT COUNT(*) FROM pricesystem_product_customer_matrix_customer
WHERE customer_id = 123;
```

Same Priority Issues

Symptom: Unpredictable pricing behavior.

Cause: Multiple matrices with same priority.

Resolution:

1. Find duplicates:

```
SELECT priority, COUNT(*) AS cnt
FROM pricesystem_product_customer_matrix
WHERE is_active = 1
GROUP BY priority
HAVING cnt > 1;
```

2. Assign unique priorities
3. Test again

Priority Not Respected

Symptom: Lower priority matrix wins.

Possible Causes:

1. High-priority matrix missing product
2. High-priority matrix expired (date range)
3. Customer-specific date override on lower-priority matrix
4. Cache issue

Debug:

1. Check product exists in all matrices
2. Verify date ranges (matrix + customer)
3. Clear cache
4. Test with fresh customer session

Advanced: Manual Priority Override

For specific customers, override priority system:

Method 1: Highest Priority Matrix

Create customer-specific matrix with priority 50+.

Example:

- Standard matrices: 10-30
- Customer exception: Priority 60
- Always wins for that customer

Method 2: Date Range Stacking

Layer matrices with different date ranges but same customer.

Example:

```
Base: Priority 10, permanent
Q1 Special: Priority 20, Jan-Mar
Q2 Special: Priority 20, Apr-Jun
VIP Override: Priority 30, permanent
```

Customer gets VIP pricing except during Q1/Q2 when specials might offer better prices (if merge enabled).

Method 3: Customer-Specific Date Restriction

Use customer-specific dates to limit when high-priority matrix applies.

Example:

```
Matrix: VIP Pricing (Priority 30, permanent at matrix level)

Customer #123:
- Override dates: 2025-01-01 to 2025-06-30 (6-month trial)
- After Jun 30: Loses VIP pricing, falls back to lower-priority matrices

Customer #456:
- No override dates
- Gets VIP pricing permanently (or until matrix expires)
```

Integration with Other Pricesystem Modules

Matrix vs Pricelist Priority

Price Resolution Order:

1. Shopping cart price rules
2. Customer-specific prices
3. **Product-Customer Matrices** ← Higher in chain
4. Named Pricelists
5. Category prices
6. Catalog price

Key Point: Matrices evaluated BEFORE pricelists. If matrix price found, pricelist never checked.

Cross-Module Conflicts

Scenario: Customer has both matrix and pricelist pricing.

Resolution:

- Matrix price found → Use it (pricelist ignored)
- Matrix price NOT found → Fallback to pricelist

Recommendation: Use matrices OR pricelists, not both, to avoid confusion.

Performance Considerations

Priority Sorting

Indexed:

```
CREATE INDEX idx_matrix_priority ON
pricesystem_product_customer_matrix(priority);
```

Query Optimization:

- Matrices sorted by priority DESC
- Highest priority evaluated first
- If merge=No, stop after first match (fast)
- If merge=Yes, evaluate all matching matrices (slower)

Merge Impact

Merge = No:

- Fast: First matching matrix wins
- Single database query per product

Merge = Yes:

- Slower: All matrices must be checked
- Multiple queries to find best price at each tier
- Consider for VIP customers only

Related Documentation

- Creating Matrices - Step-by-step matrix creation
- Matrix Pricing System - Core architecture and workflow
- Multi-Attribute Matching - Attribute logic with AND/OR