Notification Segmentation Documentation

Date: December 8, 2025
Version: 1.0
Status: Documentation Complete - Awaiting Segmentation Guidance

---

Table of Contents

1. Executive Summary
2. Notification Categories
3. Purchase Orders (PO)
4. Inventory Management
5. Payments & Invoices
6. Shipments
7. Orders
8. Calendar Events
9. Compliance
10. Current Recipient Patterns
11. Segmentation Guidance

Executive Summary

Overview

The GrowShip MVP notification system provides real-time alerts across multiple modules including Purchase Orders, Inventory, Payments, Shipments, Orders, Calendar Events, and Compliance. Notifications are created through various triggers including user actions, automated cron jobs, and database triggers.

Current State Analysis

Total Notifications Identified: 30+ distinct notification types

Current Recipient Patterns:

- **Brand Users:** Most notifications target brand admins, managers, logistics, and reviewers
- **Distributors:** Limited notifications currently sent to distributors (primarily shipment-related)
- **Super Admins:** Included in many critical notifications for visibility

- **Mixed Recipients:** Some notifications go to both brand users and distributors

Segmentation Goals

This document serves as a reference for implementing proper notification segmentation between:

- **Brand Admins/Users** - Internal brand team members
- **Distributors** - External distributor partners

Each notification below includes current recipient information and a section for guidance on proper segmentation.

Notification Categories

Purchase Orders (PO)

1. New Purchase Order Created

- **Title:** “New Purchase Order Created”
- **Type:** `order`
- **Description:** Triggered when a new PO is created and requires review
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_logistics`, `brand_reviewer` (excluding creator)
- Super Admins: All super admins (including creator if applicable)
- **Trigger Location:** `lib/notifications/po-alerts.ts` → `createPOCreatedAlert()`
- **Trigger Event:** PO creation via `hooks/use-purchase-orders.ts` → `createPOMutation`
- **Priority:** `medium`
- **Action Required:** `true`
- **Related Entity Type:** `po`
- **Action URL:** `/purchase-orders/{poId}`

Segmentation Guidance: [TO BE DETERMINED]

---

2. Purchase Order Approval Required

- **Title:** “Purchase Order Approval Required”
- **Type:** `order`
- **Description:** Triggered when a PO is submitted and requires approval
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager` (excluding submitter)
- Super Admins: All super admins (including submitter if applicable)

- **Trigger Location:** `lib/notifications/po-alerts.ts` → `createPOApprovalAlert()`
- **Trigger Event:** PO submission/status change to pending approval
- **Priority:** `high`
- **Action Required:** `true`
- **Related Entity Type:** `po`
- **Action URL:** `/purchase-orders/{poId}`

Segmentation Guidance: [TO BE DETERMINED]

---

3. Purchase Order Approved

- **Title:** "Purchase Order Approved"
- **Type:** `order`
- **Description:** Notifies PO creator when their PO is approved
- **Current Recipients:**
- PO creator only
- **Trigger Location:** `lib/notifications/po-alerts.ts` → `createPOStatusChangeAlert()`
- **Trigger Event:** PO approval via `app/api/purchase-orders/[id]/approve/route.ts`
- **Priority:** `low`
- **Action Required:** `false`
- **Related Entity Type:** `po`
- **Action URL:** `/purchase-orders/{poId}`

Segmentation Guidance: [TO BE DETERMINED]

---

4. Purchase Order Rejected

- **Title:** "Purchase Order Rejected"
- **Type:** `order`
- **Description:** Notifies PO creator when their PO is rejected
- **Current Recipients:**
- PO creator only
- **Trigger Location:** `lib/notifications/po-alerts.ts` → `createPOStatusChangeAlert()`
- **Trigger Event:** PO rejection via `app/api/purchase-orders/[id]/reject/route.ts`
- **Priority:** `medium`
- **Action Required:** `false`
- **Related Entity Type:** `po`
- **Action URL:** `/purchase-orders/{poId}`

Segmentation Guidance: [TO BE DETERMINED]

---

5. Purchase Order Placed

- **Title:** “Purchase Order Placed”
- **Type:** `order`
- **Description:** Notifies when PO status changes to “ordered”
- **Current Recipients:**
- PO creator only
- **Trigger Location:** `lib/notifications/po-alerts.ts` → `createPOStatusChangeAlert()`
- **Trigger Event:** PO status change to “ordered”
- **Priority:** `low`
- **Action Required:** `false`
- **Related Entity Type:** `po`
- **Action URL:** `/purchase-orders/{poId}`

Segmentation Guidance: [TO BE DETERMINED]

---

6. Purchase Order Received

- **Title:** “Purchase Order Received”
- **Type:** `success`
- **Description:** Notifies when PO is marked as received and inventory is updated
- **Current Recipients:**
- All approved brand users
- **Trigger Location:** `lib/inventory/po-sync.ts` → `syncPOReceipt()`
- **Trigger Event:** PO receipt processing
- **Priority:** `low`
- **Action Required:** `false`
- **Related Entity Type:** `purchase_order`
- **Action URL:** `/purchase-orders/{poId}`

Segmentation Guidance: [TO BE DETERMINED]

---

7. PO Approval Due Reminder

- **Title:** Calendar event title (e.g., “PO Approval Due”)
- **Type:** `order`
- **Description:** Reminder for upcoming PO approval deadlines
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics`
- Super Admins: All super admins
- **Trigger Location:** `lib/notifications/compliance-alerts.ts` → `checkCalendarEventAlerts()`
- **Trigger Event:** Calendar event `po_approval_due` within 3 days
- **Priority:** `high` (if due today), `medium` (if due in 1–3 days)

- **Action Required:** `true` (if due within 1 day)
- **Related Entity Type:** `calendar_event`
- **Action URL:** `/purchase-orders/{poId}`

Segmentation Guidance: [TO BE DETERMINED]

Inventory Management

8. Product Out of Stock

- **Title:** “Product Out of Stock”
- **Type:** `warning`
- **Description:** Alert when product quantity reaches zero
- **Current Recipients:**
- All approved brand users
- **Trigger Location:**
- `lib/notifications/inventory-alerts.ts` → `createLowStockAlert()` (level: “critical”)
- `lib/inventory/order-sync.ts` → `syncOrderFulfillment()` (when fulfillment causes stock to reach 0)
- **Trigger Event:**
- Automated cron check (`checkInventoryAlerts()`)
- Order fulfillment that depletes stock
- **Priority:** `urgent`
- **Action Required:** `true`
- **Related Entity Type:** `inventory`
- **Action URL:** `/products/{productId}`

Segmentation Guidance: [TO BE DETERMINED]

---

9. Critical Stock Level

- **Title:** “Critical Stock Level”
- **Type:** `warning`
- **Description:** Alert when product stock falls below critical threshold
- **Current Recipients:**
- All approved brand users
- **Trigger Location:**
- `lib/notifications/inventory-alerts.ts` → `createLowStockAlert()` (level: “critical”)
- `lib/inventory/order-sync.ts` → `syncOrderFulfillment()` (when fulfillment causes critical level)
- **Trigger Event:**
- Automated cron check (`checkInventoryAlerts()`)

- Order fulfillment that triggers critical threshold
- **Priority:** `urgent`
- **Action Required:** `true`
- **Related Entity Type:** `inventory`
- **Action URL:** `/products/{productId}`

Segmentation Guidance: [TO BE DETERMINED]

---

10. Low Stock Alert

- **Title:** “Low Stock Alert”
- **Type:** `warning`
- **Description:** Alert when product stock falls below low stock threshold
- **Current Recipients:**
- All approved brand users
- **Trigger Location:**
- `lib/notifications/inventory-alerts.ts` → `createLowStockAlert()` (level: “low”)
- `lib/inventory/order-sync.ts` → `syncOrderFulfillment()` (when fulfillment triggers low threshold)
- `lib/inventory/order-sync.ts` → `syncOrderAllocation()` (when allocation causes low available stock)
- **Trigger Event:**
- Automated cron check (`checkInventoryAlerts()`)
- Order allocation/fulfillment that triggers low threshold
- **Priority:** `high`
- **Action Required:** `true`
- **Related Entity Type:** `inventory`
- **Action URL:** `/products/{productId}`

Segmentation Guidance: [TO BE DETERMINED]

---

11. Low Available Stock

- **Title:** “Low Available Stock”
- **Type:** `warning`
- **Description:** Alert when available stock (after allocations) falls below low threshold
- **Current Recipients:**
- All approved brand users
- **Trigger Location:** `lib/inventory/order-sync.ts` → `syncOrderAllocation()`
- **Trigger Event:** Order allocation that causes available stock to fall below threshold
- **Priority:** `high`
- **Action Required:** `true`
- **Related Entity Type:** `inventory`
- **Action URL:** `/products/{productId}`

Segmentation Guidance: [TO BE DETERMINED]

---

12. Stock Running Out Soon

- **Title:** "Stock Running Out Soon"
- **Type:** `warning`
- **Description:** Predictive alert based on sales velocity (projected stock-out within 7 days)
- **Current Recipients:**
- All approved brand users
- **Trigger Location:** `lib/notifications/inventory-alerts.ts` → `createRunningOutSoonAlert()`
- **Trigger Event:** Automated cron check (`checkInventoryAlerts()`) - predictive analysis
- **Priority:** `high`
- **Action Required:** `true`
- **Related Entity Type:** `inventory`
- **Action URL:** `/products/{productId}`

Segmentation Guidance: [TO BE DETERMINED]

---

13. Stock Replenished

- **Title:** "Stock Replenished"
- **Type:** `success`
- **Description:** Notification when stock is replenished above low threshold
- **Current Recipients:**
- All approved brand users
- **Trigger Location:**
- `lib/notifications/inventory-alerts.ts` → `createStockRestoredAlert()`
- `lib/inventory/po-sync.ts` → `syncPOReceipt()` (when receipt resolves low stock)
- **Trigger Event:**
- Automated cron check detects stock restored
- PO receipt that brings stock above threshold
- **Priority:** `low`
- **Action Required:** `false`
- **Related Entity Type:** `inventory`
- **Action URL:** `/products/{productId}`

Segmentation Guidance: [TO BE DETERMINED]

---

14. Overstock Alert

- **Title:** "Overstock Alert"
- **Type:** `warning`
- **Description:** Alert when product stock exceeds maximum threshold

- **Current Recipients:**
- All approved brand users
- **Trigger Location:** `lib/notifications/inventory-alerts.ts` → `createOverstockAlert()`
- **Trigger Event:** Automated cron check (`checkInventoryAlerts()`)
- **Priority:** `medium`
- **Action Required:** `false`
- **Related Entity Type:** `inventory`
- **Action URL:** `/products/{productId}`

Segmentation Guidance: [TO BE DETERMINED]

---

15. Manual Stock Adjustment

- **Title:** “Manual Stock Adjustment”
- **Type:** `info`
- **Description:** Notification when manual stock adjustment is made
- **Current Recipients:**
- All approved brand users
- **Trigger Location:** `app/api/inventory/adjust/route.ts`
- **Trigger Event:** Manual inventory adjustment via API
- **Priority:** `low`
- **Action Required:** `false`
- **Related Entity Type:** `inventory`
- **Action URL:** `/products/{productId}`

Segmentation Guidance: [TO BE DETERMINED]

---

16. Bulk Stock Adjustment

- **Title:** “Bulk Stock Adjustment”
- **Type:** `info`
- **Description:** Summary notification for bulk stock adjustments
- **Current Recipients:**
- All approved brand users
- **Trigger Location:** `app/api/inventory/bulk-adjust/route.ts`
- **Trigger Event:** Bulk inventory adjustment via API
- **Priority:** `low`
- **Action Required:** `false`
- **Related Entity Type:** `inventory`
- **Action URL:** `/inventory/transactions?reference_number={refNumber}`

Segmentation Guidance: [TO BE DETERMINED]

Payments & Invoices

17. Payment Due Soon

- **Title:** “Payment Due Soon”
- **Type:** `payment`
- **Description:** Reminder for invoices due within 7 days
- **Current Recipients:**
- All approved brand users for the invoice’s brand
- **Trigger Location:** `lib/notifications/payment-alerts.ts` → `checkPaymentDueAlerts()`
- **Trigger Event:** Automated cron check (every 6 hours) for invoices due within 7 days
- **Priority:**
- `urgent` (≤1 day)
- `high` (≤3 days)
- `medium` (4–7 days)
- **Action Required:** `true`
- **Related Entity Type:** `invoice`
- **Action URL:** `/invoices/{invoiceId}`
- **Expires At:** Invoice due date

Segmentation Guidance: [TO BE DETERMINED]

---

18. Payment Due Reminder (Calendar Event)

- **Title:** Calendar event title (e.g., “Payment Due”)
- **Type:** `payment`
- **Description:** Reminder for upcoming payment due dates from calendar events
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics`
- Super Admins: All super admins
- **Trigger Location:** `lib/notifications/compliance-alerts.ts` → `checkCalendarEventAlerts()`
- **Trigger Event:** Calendar event `payment_due` within 3 days
- **Priority:**
- `urgent` (due today)
- `high` (due tomorrow)
- `medium` (due in 2–3 days)
- **Action Required:** `true` (if due within 1 day)
- **Related Entity Type:** `calendar_event`
- **Action URL:** `/invoices?highlight={invoiceId}`

Segmentation Guidance: [TO BE DETERMINED]

Shipments

19. Shipment Created

- **Title:** “Shipment Created”
- **Type:** `shipment`
- **Description:** Notification when a new shipment is created for an order
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics` (for order’s brand)
- Distributors: All approved distributor users (for order’s distributor)
- **Trigger Location:** Database function `create_shipment_transaction()` in `supabase_migrations/064_fix_shipment_notification_type.sql`
- **Trigger Event:** Shipment creation via RPC function
- **Priority:** `medium`
- **Action Required:** `false`
- **Related Entity Type:** `shipment`
- **Action URL:** `/orders/{orderId}`

Segmentation Guidance: [TO BE DETERMINED]

---

20. Shipment Shipped

- **Title:** “Shipment Shipped”
- **Type:** `shipment`
- **Description:** Notification when shipment status changes to “shipped”
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics` (for shipment’s brand)
- Distributors: All approved distributor users (for shipment’s distributor)
- **Trigger Location:** Database function `update_shipment_status()` in `supabase_migrations/064_fix_shipment_notification_type.sql`
- **Trigger Event:** Shipment status update to “shipped”
- **Priority:** `medium`
- **Action Required:** `false`
- **Related Entity Type:** `shipment`
- **Action URL:** `/orders/{orderId}`

Segmentation Guidance: [TO BE DETERMINED]

---

21. Shipment In Transit

- **Title:** “Shipment In Transit”
- **Type:** `shipment`
- **Description:** Notification when shipment status changes to “in_transit”
- **Current Recipients:**

- Brand users: `brand_admin`, `brand_manager`, `brand_logistics` (for shipment's brand)
- Distributors: All approved distributor users (for shipment's distributor)
- **Trigger Location:** Database function `update_shipment_status()` in `supabase_migrations/064_fix_shipment_notification_type.sql`
- **Trigger Event:** Shipment status update to "in_transit"
- **Priority:** `medium`
- **Action Required:** `false`
- **Related Entity Type:** `shipment`
- **Action URL:** `/orders/{orderId}`

Segmentation Guidance: [TO BE DETERMINED]

---

22. Shipment Delivered

- **Title:** "Shipment Delivered"
- **Type:** `shipment`
- **Description:** Notification when shipment status changes to "delivered"
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics` (for shipment's brand)
- Distributors: All approved distributor users (for shipment's distributor)
- **Trigger Location:** Database function `update_shipment_status()` in `supabase_migrations/064_fix_shipment_notification_type.sql`
- **Trigger Event:** Shipment status update to "delivered"
- **Priority:** `low`
- **Action Required:** `false`
- **Related Entity Type:** `shipment`
- **Action URL:** `/orders/{orderId}`

Segmentation Guidance: [TO BE DETERMINED]

---

23. Shipment Cancelled

- **Title:** "Shipment Cancelled"
- **Type:** `shipment`
- **Description:** Notification when shipment status changes to "cancelled"
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics` (for shipment's brand)
- Distributors: All approved distributor users (for shipment's distributor)
- **Trigger Location:** Database function `update_shipment_status()` in `supabase_migrations/064_fix_shipment_notification_type.sql`
- **Trigger Event:** Shipment status update to "cancelled"
- **Priority:** `medium`
- **Action Required:** `false`
- **Related Entity Type:** `shipment`
- **Action URL:** `/orders/{orderId}`

Segmentation Guidance: [TO BE DETERMINED]

---

24. Shipment Failed

- **Title:** “Shipment Failed”
- **Type:** `shipment`
- **Description:** Notification when shipment status changes to “failed”
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics` (for shipment’s brand)
- Distributors: All approved distributor users (for shipment’s distributor)
- **Trigger Location:** Database function `update_shipment_status()` in `supabase_migrations/064_fix_shipment_notification_type.sql`
- **Trigger Event:** Shipment status update to “failed”
- **Priority:** `high`
- **Action Required:** `false`
- **Related Entity Type:** `shipment`
- **Action URL:** `/orders/{orderId}`

Segmentation Guidance: [TO BE DETERMINED]

---

25. Shipment Arrival Reminder (Calendar Event)

- **Title:** Calendar event title (e.g., “Shipment Arrival”)
- **Type:** `shipping`
- **Description:** Reminder for upcoming shipment arrivals from calendar events
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics`
- Super Admins: All super admins
- **Trigger Location:** `lib/notifications/compliance-alerts.ts` → `checkCalendarEventAlerts()`
- **Trigger Event:** Calendar event `shipment_arrival` or `delivery_milestone` within 3 days
- **Priority:** `medium` (if due today), `low` (if due in 1–3 days)
- **Action Required:** `false`
- **Related Entity Type:** `calendar_event`
- **Action URL:** `/purchase-orders/{poId}` or `/shipments?highlight={shipmentId}`

Segmentation Guidance: [TO BE DETERMINED]

Orders

26. Order Fulfilled

- **Title:** “Order Fulfilled”
- **Type:** `success`
- **Description:** Notification when order is fulfilled and inventory is consumed
- **Current Recipients:**
- All approved brand users
- **Trigger Location:** `lib/inventory/order-sync.ts` → `syncOrderFulfillment()`
- **Trigger Event:** Order fulfillment processing
- **Priority:** `low`
- **Action Required:** `false`
- **Related Entity Type:** `order`
- **Action URL:** `/orders/{orderId}`

Segmentation Guidance: [TO BE DETERMINED]

---

27. Order Cancelled

- **Title:** “Order Cancelled”
- **Type:** `info`
- **Description:** Notification when order is cancelled and allocated stock is released
- **Current Recipients:**
- All approved brand users
- **Trigger Location:** `lib/inventory/order-sync.ts` → `syncOrderCancellation()`
- **Trigger Event:** Order cancellation processing
- **Priority:** `low`
- **Action Required:** `false`
- **Related Entity Type:** `order`
- **Action URL:** `/orders/{orderId}`

Segmentation Guidance: [TO BE DETERMINED]

---

28. Backorder Review Reminder (Calendar Event)

- **Title:** Calendar event title (e.g., “Backorder Review”)
- **Type:** `inventory`
- **Description:** Reminder for backorder review events
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics`
- Super Admins: All super admins
- **Trigger Location:** `lib/notifications/compliance-alerts.ts` → `checkCalendarEventAlerts()`
- **Trigger Event:** Calendar event `backorder_review` within 3 days
- **Priority:** `medium`
- **Action Required:** `false`
- **Related Entity Type:** `calendar_event`
- **Action URL:** `/purchase-orders/{poId}`

Segmentation Guidance: [TO BE DETERMINED]

Calendar Events

29. Custom Event Reminder

- **Title:** Calendar event title
- **Type:** `reminder`
- **Description:** Generic reminder for custom calendar events
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics`
- Super Admins: All super admins
- **Trigger Location:** `lib/notifications/compliance-alerts.ts` → `checkCalendarEventAlerts()`
- **Trigger Event:** Any calendar event (excluding specific types) within 3 days
- **Priority:** `low`
- **Action Required:** `false` (unless due within 1 day)
- **Related Entity Type:** `calendar_event`
- **Action URL:** `/calendar`

Segmentation Guidance: [TO BE DETERMINED]

Compliance

30. Compliance Review Due

- **Title:** “Compliance Review Due Today/Tomorrow/in X days”
- **Type:** `warning`
- **Description:** Alert for upcoming monthly distributor compliance review deadlines
- **Current Recipients:**
- Brand users: `brand_admin`, `brand_manager`, `brand_logistics`
- Super Admins: All super admins
- **Trigger Location:** `lib/notifications/compliance-alerts.ts` → `checkComplianceAlerts()`
- **Trigger Event:** Calendar event `compliance_review` within 3 days
- **Priority:**
- `urgent` (due today)
- `high` (due tomorrow)
- `medium` (due in 2–3 days)
- **Action Required:** `true`
- **Related Entity Type:** `calendar_event`
- **Action URL:** `/calendar`

Segmentation Guidance: [TO BE DETERMINED]

Current Recipient Patterns

Brand Users Only

Most notifications currently target brand users with specific roles:

- **brand_admin** - Brand administrators
- **brand_manager** - Brand managers
- **brand_logistics** - Logistics coordinators
- **brand_reviewer** - PO reviewers

Notifications in this category:

- All Purchase Order notifications (except creator-specific)
- All Inventory Management notifications
- Payment Due alerts
- Order fulfillment/cancellation
- Calendar event reminders
- Compliance reviews

Distributors Only

Currently, distributors receive very few notifications:

- None explicitly identified (shipment notifications go to both brand and distributors)

Super Admins

Super admins are included in many critical notifications for platform-wide visibility:

- PO creation and approval requests
- Calendar event reminders
- Compliance reviews

Mixed Recipients (Brand + Distributors)

Shipment-related notifications are sent to both:

- Brand users (admin, manager, logistics roles)

- Distributor users (all approved users for the order's distributor)

Notifications in this category:

- Shipment Created
- Shipment Shipped
- Shipment In Transit
- Shipment Delivered
- Shipment Cancelled
- Shipment Failed

Segmentation Guidance

Purpose

This section will be populated with guidance on which notifications should be sent to Brand Admins vs Distributors after review.

Current Questions to Address

1. **Purchase Orders:**
2. Should distributors be notified when POs are created for their orders?
3. Should distributors see PO approval/rejection status?
4. Should distributors be notified when POs are received?
5. **Inventory:**
6. Should distributors receive stock alerts for products they distribute?
7. Should distributors see stock replenishment notifications?
8. Should distributors be notified of manual adjustments?
9. **Payments:**
10. Should distributors receive payment due reminders?
11. Should distributors see invoice-related notifications?
12. **Shipments:**
13. Current: Both brand and distributors receive shipment notifications
14. Question: Should this remain the same or be segmented differently?
15. **Orders:**
16. Should distributors be notified when orders are fulfilled?
17. Should distributors be notified when orders are cancelled?
18. **Calendar Events:**
19. Should distributors receive calendar reminders?
20. Which calendar events are relevant to distributors?
21. **Compliance:**
22. Should distributors receive compliance review reminders?
23. Should distributors be notified about their own compliance deadlines?

Implementation Notes

Once segmentation guidance is provided, the following changes will be needed:

1. **Update Notification Functions:**
2. Modify `createNotificationsForBrand()` to support distributor filtering
3. Create `createNotificationsForDistributor()` function
4. Update recipient queries in all notification functions
5. **Database Queries:**
6. Update user profile queries to filter by `distributor_id` where needed
7. Ensure proper role-based filtering
8. **Shipment Notifications:**
9. Review database functions (`create_shipment_transaction`, `update_shipment_status`)
10. May need to split into separate brand/distributor notification calls
11. **Testing:**
12. Test notifications for brand users
13. Test notifications for distributors
14. Verify no cross-contamination between brands/distributors

---

Summary Statistics

Category	Total Notifications	Brand Only	Distributor Only	Mixed	Super Admin Included
Purchase Orders	7	6	0	0	2
Inventory	9	9	0	0	0
Payments	2	2	0	0	0
Shipments	7	0	0	7	0
Orders	3	3	0	0	0
Calendar Events	1	1	0	0	1
Compliance	1	1	0	0	1
TOTAL	**30**	**22**	**0**	**7**	**4**

Next Steps

1. **Review this document** and provide guidance on segmentation
2. **Identify which notifications** should go to Brand Admins vs Distributors
3. **Implement segmentation logic** based on guidance
4. **Test notification delivery** for both user types
5. **Update documentation** with final segmentation rules

Document Version: 1.0
Last Updated: December 8, 2025
Maintained By: GrowShip Development Team