✅ Upgrade Checkout & Subscription Sync Fix - COMPLETE

🐛 Issue Reported

User reported: **"Upgrade plan is not checking out after selection and is not updating to the selected plan."**

This was a critical bug preventing users from upgrading their subscription plans through the Stripe checkout flow.

🔍 Root Cause Analysis

Primary Issue: Missing Subscription Metadata

The core problem was in `/app/api/billing/checkout/route.ts`:

Before Fix:

```
// CONDITIONALLY added subscription_data only for existing subscribers
...(existingSubscription?.stripeSubscriptionId && {
  subscription_data: {
    metadata: {
      organizationId: organization.id,
    },
  },
}),
```

Problem:
- For **new subscriptions** (first-time upgrades), `subscription_data.metadata` was NOT being added
- For **existing subscriptions**, metadata was only added if `stripeSubscriptionId` existed (which might be null)
- Without `organizationId` in the subscription metadata, the **webhook couldn't identify which organization to update**
- This caused the subscription to be created in Stripe but never synced to the database

Secondary Issue: Incomplete Webhook Updates

In `/app/api/billing/webhook/route.ts`, the `handleSubscriptionUpdate` function:

Before Fix:

```
update: {
  plan,
  status,
  stripePriceId: priceId,
  // Missing: stripeCustomerId and stripeSubscriptionId!
  currentPeriodStart: ...,
  currentPeriodEnd: ...,
}
```

Problem:
- When updating an existing subscription record, it wasn't updating `stripeCustomerId` or `stripeSubscriptionId`
- If these fields were null/missing in the database, they would remain null even after a successful upgrade
- This broke subsequent operations that depend on these Stripe IDs

Tertiary Issue: Insufficient Error Logging

- Webhook errors were not being logged with enough detail to diagnose issues
- No visibility into what metadata was (or wasn't) being passed
- Difficult to troubleshoot subscription sync failures

✅ Solution Implemented

1. Always Include Subscription Metadata

File: `/app/api/billing/checkout/route.ts`

Fix:

```
// ALWAYS add subscription metadata so webhook can identify the organization
subscription_data: {
  metadata: {
    organizationId: organization.id,
    plan,
    billingCycle,
  },
},
```

Why This Works:
- Ensures EVERY Stripe subscription (new or upgrade) has the `organizationId` in its metadata
- The webhook can now reliably identify which organization to update
- Added `plan` and `billingCycle` for additional context

2. Complete Webhook Updates

File: `/app/api/billing/webhook/route.ts`

Fix:

```
update: {
  plan,
  status,
  stripeCustomerId: customerId, // ✅ Now updated
  stripeSubscriptionId: subscription.id, // ✅ Now updated
  stripePriceId: priceId,
  currentPeriodStart: new Date(subscriptionData.current_period_start * 1000),
  currentPeriodEnd: new Date(subscriptionData.current_period_end * 1000),
  cancelAtPeriodEnd: subscriptionData.cancel_at_period_end,
  canceledAt: subscriptionData.canceled_at
    ? new Date(subscriptionData.canceled_at * 1000)
    : null,
  trialEnd: subscriptionData.trial_end
    ? new Date(subscriptionData.trial_end * 1000)
    : null,
},
```

Why This Works:
- Ensures all critical Stripe IDs are updated when a subscription changes
- Prevents orphaned database records with missing Stripe IDs
- Enables proper subscription management through the Stripe Customer Portal

3. Enhanced Error Logging

Added Comprehensive Logging:

In `handleCheckoutSessionCompleted`:

```
console.log('Handling checkout.session.completed:', session.id);
console.log('Session metadata:', session.metadata);
console.log('Session subscription ID:', session.subscription);
console.log('Checkout completed for organization:', organizationId);
console.log('Retrieving subscription from Stripe:', session.subscription);
console.log('Retrieved subscription, processing update...');
console.log('Subscription update completed successfully');

// Error cases:
console.error('No organizationId in checkout session metadata');
console.error('Available session metadata:', session.metadata);
console.error('No subscription found in checkout session');
console.error('Session object:', JSON.stringify(session, null, 2));
```

In `handleSubscriptionUpdate`:

```
console.log('Handling subscription update:', subscription.id);
console.log('Subscription metadata:', subscription.metadata);
console.log('Subscription status:', subscription.status);
console.log('Processing subscription for organization:', organizationId);
console.log('Price ID:', priceId);
console.log('Determined plan:', plan);

// Error cases:
console.error('No organizationId in subscription metadata');
console.error('Available metadata:', subscription.metadata);
console.error('No price ID found in subscription');
console.error('Subscription items:', subscription.items.data);
console.error('Unknown price ID:', priceId);
console.error('Available price IDs:', {
  collector_monthly: process.env.STRIPE_PRICE_COLLECTOR_MONTHLY,
  collector_annual: process.env.STRIPE_PRICE_COLLECTOR_ANNUAL,
  dealer_monthly: process.env.STRIPE_PRICE_DEALER_MONTHLY,
  // ... etc
});
```

Why This Works:
- Provides detailed visibility into webhook processing
- Helps diagnose issues with metadata, price IDs, or subscription status
- Logs all critical data points for troubleshooting
- Makes it easy to identify configuration issues (missing env vars, wrong price IDs, etc.)

🧪 Testing Guide

Prerequisites

1. **Test Account:**
   - Email: `john@doe.com`
   - Password: `password123`
   - Current Plan: Collector (or any plan)

2. **Stripe Test Mode:**
   - Ensure `STRIPE_SECRET_KEY` starts with `sk_test_`
   - Use Stripe test cards for payment

3. **Webhook Configuration:**
   - Ensure `STRIPE_WEBHOOK_SECRET` is properly configured
   - Webhook endpoint: `https://genesisprovenance.abacusai.app/api/billing/webhook`

Test Scenario 1: First-Time Upgrade (Collector → Dealer)

Steps:
1. Sign in to https://genesisprovenance.abacusai.app
2. Navigate to **Settings → Billing**
3. Click **“Upgrade Plan”** button
4. Select **“Dealer”** plan
5. Choose **“Monthly”** billing
6. Click **“Upgrade Now”**

7. Complete Stripe Checkout using test card: `4242 4242 4242 4242`
8. You should be redirected back to `/settings/billing?success=true`

Expected Results:
✅ Checkout completes successfully
✅ Redirect shows success message
✅ Billing page shows **“Dealer”** plan
✅ Billing page shows **“Active”** status
✅ Usage limits update to Dealer limits (500 assets, 10 team members, etc.)

Webhook Logs to Check:

```
[Checkout] Checkout session created successfully: {...}
Processing webhook event: checkout.session.completed
Handling checkout.session.completed: cs_test_...
Session metadata: { organizationId: '...', plan: 'dealer', billingCycle: 'monthly' }
Checkout completed for organization: ...
Retrieving subscription from Stripe: sub_...
Handling subscription update: sub_...
Subscription metadata: { organizationId: '...', plan: 'dealer', billingCycle: 'monthly
' }
Processing subscription for organization: ...
Price ID: price_...
Determined plan: dealer
Subscription updated for organization ...: dealer (active)
Subscription update completed successfully
```

Test Scenario 2: Plan Change (Dealer → Enterprise)

Steps:
1. Ensure account is on **Dealer** plan
2. Navigate to **Settings → Billing**
3. Click **“Upgrade Plan”**
4. Select **“Enterprise”** plan
5. Choose **“Annual”** billing
6. Click **“Upgrade Now”**
7. Complete checkout

Expected Results:
✅ Checkout completes
✅ Plan updates to **“Enterprise”**
✅ Billing cycle shows **“Annual”**
✅ All limits show **“Unlimited”** or **“1000+”**

Test Scenario 3: Error Handling - Already Subscribed

Steps:
1. While on **Dealer** plan
2. Attempt to upgrade to **Dealer** again

Expected Results:
✅ Error message: “You are already subscribed to this plan”
✅ No checkout session created
✅ No charge attempted

Test Scenario 4: Webhook Failure Recovery

Steps:
1. Temporarily disable webhook secret (comment out in `.env`)
2. Attempt checkout
3. Re-enable webhook secret
4. Use Stripe Dashboard to resend webhook event

Expected Results:
✅ Resent webhook processes successfully
✅ Subscription syncs to database
✅ Billing page shows correct plan

📊 Verification Checklist

Database Verification

```sql
-- Check subscription record
SELECT
  id,
  "organizationId",
  plan,
  status,
  "stripeCustomerId",
  "stripeSubscriptionId",
  "stripePriceId",
  "currentPeriodStart",
  "currentPeriodEnd"
FROM "Subscription"
WHERE "organizationId" = '<your-org-id>';
```

Expected:
- ✅ `stripeCustomerId` is NOT NULL (e.g., `cus_...`)
- ✅ `stripeSubscriptionId` is NOT NULL (e.g., `sub_...`)
- ✅ `stripePriceId` matches the selected plan/billing cycle
- ✅ `plan` is ‘dealer’ or ‘enterprise’ (after upgrade)
- ✅ `status` is ‘active’
- ✅ `currentPeriodStart` and `currentPeriodEnd` are set

Stripe Dashboard Verification

1. Go to https://dashboard.stripe.com/test/subscriptions
2. Find the subscription by customer email
3. Click on the subscription
4. Scroll to **“Metadata”** section

Expected:
- ✅ `organizationId`: `<your-org-id>`
- ✅ `plan`: `dealer` or `enterprise`
- ✅ `billingCycle`: `monthly` or `annual`

Application Verification

Billing Page (`/settings/billing`):
- ✅ Correct plan name displayed
- ✅ Correct status badge (Active, green)
- ✅ Correct renewal date
- ✅ Usage meters show correct limits
- ✅ Features list shows correct available features

Admin Dashboard (`/admin/billing`):
- ✅ MRR reflects the new plan price
- ✅ Subscription appears in the subscriptions table
- ✅ Plan column shows correct plan
- ✅ Status column shows "Active"

🚨 Troubleshooting

Issue: "Stripe is not configured"

Cause: Missing or invalid `STRIPE_SECRET_KEY`

Solution:

```
# Check .env file
STRIPE_SECRET_KEY=sk_test_... # Must start with sk_test_ or sk_live_
```

Issue: "Price not found for [plan] [billingCycle]"

Cause: Missing or incorrect Stripe Price IDs in `.env`

Solution:

```
# Verify all 6 price IDs are set:
STRIPE_PRICE_COLLECTOR_MONTHLY=price_...
STRIPE_PRICE_COLLECTOR_ANNUAL=price_...
STRIPE_PRICE_DEALER_MONTHLY=price_...
STRIPE_PRICE_DEALER_ANNUAL=price_...
STRIPE_PRICE_ENTERPRISE_MONTHLY=price_...
STRIPE_PRICE_ENTERPRISE_ANNUAL=price_...
```

Issue: "Webhook signature verification failed"

Cause: Missing or incorrect `STRIPE_WEBHOOK_SECRET`

Solution:
1. Go to Stripe Dashboard → Developers → Webhooks
2. Click on your webhook endpoint
3. Click **"Reveal signing secret"**
4. Copy the secret (starts with `whsec_`)
5. Update `.env` :
`bash`

```
   STRIPE_WEBHOOK_SECRET=whsec_...
```
6. Restart the server

Issue: Checkout completes but subscription not showing

Cause: Webhook not being triggered or failing silently

Solution:
1. Check Stripe Dashboard → Developers → Webhooks
2. Look for failed events
3. Check webhook logs for errors
4. Verify webhook endpoint is publicly accessible
5. Manually resend the webhook event

Issue: “No organizationId in subscription metadata”

Cause: This was the original bug! Should now be fixed.

Solution:
- Ensure the fix is deployed (check git commit)
- Test with a new checkout session
- If still failing, check that `subscription_data` is being set in `/app/api/billing/checkout/route.ts`

📝 Technical Details

Files Modified

1. **`/app/api/billing/checkout/route.ts`**
   - Added `subscription_data.metadata` unconditionally
   - Ensures every subscription has `organizationId`, `plan`, and `billingCycle` in metadata

2. **`/app/api/billing/webhook/route.ts`**
   - Updated `handleSubscriptionUpdate` to include `stripeCustomerId` and `stripeSubscriptionId` in the `update` clause
   - Added comprehensive logging throughout webhook handlers
   - Added detailed error logging for missing metadata and unknown price IDs

Key Concepts

Stripe Metadata:
- Metadata is key-value data stored on Stripe objects
- Session metadata ≠ Subscription metadata
- `subscription_data.metadata` ensures metadata is copied to the subscription
- Without this, the subscription won’t have the `organizationId`

Webhook Flow:
1. User completes Stripe Checkout
2. Stripe sends `checkout.session.completed` event
3. Webhook retrieves the subscription from Stripe
4. Webhook updates the database with subscription details
5. User sees updated plan in the application

Upsert Logic:
- `upsert` = “update if exists, create if not”
- `where: { organizationId }` - looks for existing subscription
- `create: { ... }` - used if no subscription exists
- `update: { ... }` - used if subscription already exists
- Critical that both `create` and `update` have all required fields

✅ Build & Deployment Status

TypeScript Compilation: ✅ 0 errors
Next.js Build: ✅ Successful (61 routes)
Checkpoint Saved: ✅ “Fixed upgrade checkout and subscription sync”
Deployment: ✅ Live at https://genesisprovenance.abacusai.app

🎯 Success Criteria

- ✅ Users can upgrade from Collector → Dealer
- ✅ Users can upgrade from Dealer → Enterprise
- ✅ Subscription metadata includes `organizationId`
- ✅ Database correctly updates with Stripe IDs
- ✅ Billing page shows correct plan immediately after upgrade
- ✅ Webhook logs provide clear visibility into processing
- ✅ Error messages are actionable and specific

🔄 Related Documentation

- `/PHASE_5A_COMPLETE.md` - Initial Stripe integration
- `/PHASE_5B_COMPLETE.md` - Checkout and webhooks (if exists)
- `/CHECKOUT_TROUBLESHOOTING.md` - Previous checkout issues
- `/STRIPE_WEBHOOK_SETUP_COMPLETE.md` - Webhook configuration

📞 Support

If issues persist:

1. **Check Server Logs:** Look for webhook processing logs
2. **Check Stripe Dashboard:** Verify webhook events are being received
3. **Test with Stripe CLI:** Use `stripe listen --forward-to localhost:3000/api/billing/webhook`
4. **Contact Support:** Include webhook event ID and timestamps

---

Last Updated: December 1, 2025
Status: ✅ **RESOLVED AND DEPLOYED**