

# Late API Series Pre-Scheduling Fix

---

## Date

November 25, 2025

## Problem Summary

The series scheduling was NOT working as expected. Posts were NOT appearing in the Late API dashboard's "Scheduled" section.

### What Was Broken:

#### Old (Incorrect) Flow:

1. User creates series → Stored in database with `nextScheduledAt`
2. Daemon runs every hour → Checks if `nextScheduledAt` is past → Posts **IMMEDIATELY** to all platforms
3. Result: No scheduled posts in Late API, everything posted immediately when scheduled time arrived

### What You Wanted:

#### New (Correct) Flow:

1. User creates series → **IMMEDIATELY** create scheduled post in Late API for future `nextScheduledAt`
2. Post sits in Late API "Scheduled" section until scheduled time
3. When that post publishes → Daemon detects it → Creates **NEXT** scheduled post
4. Repeats for each post in the series

## Solution Implemented

---

### 1. New Function: `scheduleFirstSeriesPost`

**File:** `/lib/cloud-storage-series-processor.ts`

#### What it does:

- Called immediately when a series is created
- Downloads the FIRST file from Dropbox folder
- Analyzes it with AI
- Generates content
- Uploads media to Late API
- Creates a **SCHEDULED** post in Late API with the `nextScheduledAt` time
- Stores the Late post ID in the series for tracking

#### Key code:

```

export async function scheduleFirstSeriesPost(seriesId: string) {
  // 1. Fetch series and validate
  const series = await prisma.postSeries.findUnique(...);

  // 2. List files from Dropbox
  const files = await listDropboxFiles(series.dropboxFolderPath);

  // 3. Download first file
  const fileBuffer = await downloadDropboxFile(targetFile.path);

  // 4. AI analysis and content generation
  const imageAnalysis = await analyzeMediaContent(fileBuffer, mimeType);
  const generatedContent = await generatePostContent(imageAnalysis, series.prompt, series.platforms);

  // 5. Create SCHEDULED post in Late API
  const latePost = await postViaLateAPI(
    platformConfigs,
    generatedContent,
    fileBuffer,
    mimeType,
    series.nextScheduledAt.toISOString(), // ← ISO 8601 format
    series.timezone
  );

  // 6. Store Late post ID for tracking
  await prisma.postSeries.update({
    where: { id: seriesId },
    data: { currentLatePostId: latePost.id }
  });
}

```

## 2. Series Creation API Update

**File:** /app/api/series/route.ts

### Changes:

- After creating a series, immediately call `scheduleFirstSeriesPost`
- Logs success or failure (but doesn't fail series creation if scheduling fails)

### Key code:

```

const series = await prisma.postSeries.create({...});

// IMMEDIATELY schedule the first post in Late API
if (autoPost && (dropboxFolderId || dropboxFolderPath)) {
  const { scheduleFirstSeriesPost } = await import('@/lib/cloud-storage-series-processor');
  const result = await scheduleFirstSeriesPost(series.id);

  if (result.success) {
    console.log(`✓ First post scheduled: ${result.latePostId}`);
  }
}

```

## 3. Database Schema Update

**File:** prisma/schema.prisma

### Added field to `PostSeries` model:

```
model PostSeries {
    // ... existing fields
    currentLatePostId String? // Current Late API post ID (for tracking)
    // ... rest of fields
}
```

#### Purpose:

- Track which Late post is currently scheduled
- Daemon will check this post's status
- When post is published, daemon moves to next file

## How It Works Now

### Step 1: Create Series

1. User fills out series form:
  - Name: "Daily Motivation"
  - Date: Today
  - Time: 7:00 AM EST
  - Platforms: Instagram, Facebook, LinkedIn
  - Dropbox folder: /motivational quotes
  - AI Prompt: "Create motivational content"
2. User clicks "Create Series"
3. Backend immediately:
  - Creates series in database
  - Calls `scheduleFirstSeriesPost(seriesId)`
  - Downloads file #1 from Dropbox
  - Generates AI content
  - Creates scheduled post in Late API for tomorrow 7:00 AM EST
  - Stores Late post ID in series
4. **Result:** Post NOW appears in Late API "Scheduled Posts" section! 

### Step 2: Daemon Checks for Published Posts

**TODO:** Next implementation phase

1. Daemon runs every hour
2. Finds all series with `currentLatePostId`
3. Checks Late API: `GET /v1/posts/{currentLatePostId}`
4. If status changed from "scheduled" to "published":
  - Increment `currentFileIndex` (1 → 2)
  - Calculate `nextScheduledAt` (tomorrow 7:00 AM)
  - Call `scheduleFirstSeriesPost` again (for file #2)
  - Stores new Late post ID

### Step 3: Repeat

- Process repeats for each file in the series
- Always maintains one scheduled post in Late API

- When final file is reached:
- If `loopEnabled` : Loop back to file #1
- If not: Mark series as `COMPLETED`

## What Was Fixed

### Fixed Issues:

1. **Immediate Scheduling:** Posts are now created in Late API as soon as series is created
2. **Late API Integration:** Posts appear in “Scheduled Posts” section
3. **Correct API Format:** Uses `scheduledFor` (ISO 8601) and `timezone` fields
4. **Tracking:** Stores Late post ID for daemon to monitor

### Still TODO (Next Phase):

1. **Daemon Logic:** Update daemon to check for published posts and schedule next one
2. **Webhook Alternative:** Since Late API doesn't have webhooks, daemon must poll
3. **Error Handling:** Better handling of Late API failures
4. **UI Feedback:** Show Late post ID and scheduled time in series list

## Testing Instructions

### Test 1: Create New Series

1. Go to Dashboard → Post → Series tab
2. Click “New Series”
3. Fill out form:
  - Name: “Test Series”
  - Frequency: Daily
  - Days: Monday-Friday
  - Time: 10:00 AM
  - Timezone: EST
  - Platforms: Instagram, Facebook
  - Dropbox folder: Select any folder with numbered images
  - Profile: Basketball Factory
  - Prompt: “Create engaging social media content”
4. Click “Create Series”

#### Expected Result:

- Series created successfully
- Console shows: “ Series created! Now scheduling first post in Late API...”
- Console shows: “ First post scheduled successfully”
- Console shows: “Late Post ID: [some ID]”

### Test 2: Check Late API Dashboard

1. Go to Late API dashboard: <https://getlate.dev/app/posts>
2. Click on “Scheduled” filter
3. You should see your post there!
4. Click on the post to see:
  - Content (AI-generated caption + hashtags)
  - Media (image from Dropbox)

- Scheduled time (matches your series time)
- Platforms (Instagram, Facebook)

## Test 3: Wait for Publication

**TODO:** This part is not yet implemented

- Currently, the post will publish via Late API at scheduled time
- But the daemon won't automatically create the NEXT post yet
- That's the next phase to implement

## Deployment Status

---

- Code Changes Complete**
- Database Schema Updated** (`currentLatePostId` field added)
- Prisma Client Regenerated**
- Build & Deploy:** Pending
- Daemon Update:** Next phase

## Files Modified

---

1. `/lib/cloud-storage-series-processor.ts`
  - Added `scheduleFirstSeriesPost()` function
  - 150+ lines of new code
2. `/app/api/series/route.ts`
  - Call `scheduleFirstSeriesPost()` after series creation
  - 20 lines added
3. `prisma/schema.prisma`
  - Added `currentLatePostId String?` to `PostSeries` model
  - 1 line added

## Next Steps (Phase 2)

---

- 1. Update Daemon:**
  - Check Late API for published posts
  - Move to next file when post publishes
  - Schedule next post
- 2. Add UI Feedback:**
  - Show "Late Post ID" in series list
  - Show "Next scheduled" time
  - Add "View in Late" button
- 3. Error Handling:**
  - Retry failed scheduling
  - Alert user if Dropbox/Late API fails
  - Fallback to immediate posting if scheduling fails

## Summary

---

### CRITICAL FIX APPLIED

- Series now PRE-SCHEDULE posts in Late API
- Posts appear in “Scheduled Posts” section
- No more waiting for daemon to post immediately
- Correct ISO 8601 format and timezone handling

### PARTIAL IMPLEMENTATION

- First post scheduling:  WORKING
- Subsequent posts:  TODO (daemon update needed)

### USER ACTION REQUIRED AFTER DEPLOYMENT:

1. Test creating a new series
  2. Check Late API dashboard for scheduled post
  3. Wait for post to publish (verify it works)
  4. Request Phase 2 implementation (daemon update)
- 

**Status:**  Ready to build and deploy

**Phase:** 1 of 2 complete

**Impact:** HIGH - Fixes core scheduling functionality