ChatGPT

Seedream (WaveSpeed AI) Integration Guide for Preset

This document explains how to integrate **Bytedance Seedream V4** via **WaveSpeed AI** alongside the existing NanoBanana integration. Seedream is a multi-modal generative AI model that can create images from text, edit existing images and even generate group pictures. It offers higher quality, faster rendering and more editing capabilities compared to NanoBanana [1]. The guide covers the model's features, authentication, API endpoints and how to hook it into your credit management and callback infrastructure.

1. Overview of Seedream

1.1 Features

Seedream 4.0 is described as a **state-of-the-art image model** that *surpasses NanoBanana in every aspect* [1]. Key advantages include:

- **Multi-modal generation support**: Seedream can handle text-to-image, image-editing and group image generation in one model [1].
- **Precise instruction editing** and **high feature retention**: the model follows instructions closely while maintaining critical image features [1].
- **Deep understanding**: it is better at interpreting context and user intent [1].
- **Ultra-fast inference**: generating a 2K image can take as little as **1.8 seconds** [1].
- **High-resolution outputs**: 2K (2048×2048) images are supported [2].
- **Rich editing functions**: element addition/deletion, frame/brush editing, style and texture changes, structural editing (e.g. swapping or modifying objects), and feature editing such as colour tone or attribute changes [1].
- **Diverse scene applications**: commercial design (posters, packaging, e-commerce), entertainment (anime/film roles), fine art, architectural design and more [1].

1.2 Prompt writing and guidance

- Use clear prompts following `"action" + "change object" + "target feature"` patterns to achieve desired results [1].
- When generating **multiple images**, use group prompts or scene descriptors to maintain consistency [1].
- Seedream docs recommend a **problem self-check list**: ensure instructions are complete, avoid missing details or redundant statements, and adjust prompts if results are incorrect [1].

2. Authentication

WaveSpeed AI uses a **Bearer token**. Obtain a WaveSpeed API key from your WaveSpeedAI dashboard. The API key must be passed in the `Authorization` header for each request. The docs reference an Authentication Guide, but the main requirement is:

```
Authorization: Bearer <WAVESPEED_API_KEY>
```

You should store this key in an environment variable (`WAVESPEED_API_KEY`) and never expose it to client-side code.

3. API Endpoints

WaveSpeed provides endpoints for submitting a task and fetching results. The path differs based on the model variant (e.g. Seedream V4, Seedream V4 Edit). Example cURL snippets from the docs show how to make a request:

```
# Submit the task (POST)
curl --location --request POST "https://api.wavespeed.ai/api/v3/bytedance/
seedream-v4" \
  --header "Authorization: Bearer $WAVESPEED_API_KEY" \
  --header "Content-Type: application/json" \
  --data '{
    "prompt": "cinematic portrait of a dancer in foggy light",
    "num_images": 2,
    "callback_url": "https://your-app.com/api/imagegen/callback",
    "size": "2048x2048",
    "enable_sync_mode": false
  }'

# Get task result (GET)
curl --location --request GET "https://api.wavespeed.ai/api/v3/predictions/
<taskId>" \
  --header "Authorization: Bearer $WAVESPEED_API_KEY"
```

3.1 Core request parameters

- **prompt** (string, required): description of the desired image or edit instructions.
- **num_images** (integer, optional): number of images to generate (similar to `numImages` in NanoBanana). Seedream may support up to 4 images in one request.
- **image_urls** (array, required only for editing): list of existing image URLs when performing image editing.
- **callback_url** (string, required): your server's callback endpoint to receive asynchronous notifications when a task is done.

- **size** (string): resolution such as `"2048x2048"` [2]; Seedream supports high resolution.
- **enable_sync_mode** (boolean): set to false for asynchronous operation; keep consistent with your existing async model.
- Additional knobs (optional): guidance scale, seed value, style presets etc. These can be passed via an `extras` object and are model specific.

3.2 Response

- When a task is created successfully, WaveSpeed returns an object containing a `task_id`. Save this ID in your `enhancement_tasks` table.
- Polling the predictions endpoint or receiving a callback will yield a payload with a `status` field (`queued`, `processing`, `succeeded`, `failed`) and an `outputs` array containing URLs of generated images.
- Use `error_code` and `error_message` to handle failures.

4. Webhooks

WaveSpeed sends task completion events to your `callback_url`. The docs emphasise a **webhook verification** mechanism: signatures in HTTP headers must be verified using the secret provided in your WaveSpeed account [2]. Implement a verification function similar to your NanoBanana webhook logic:

1. Read the signature header and request body.
2. Use the shared secret to compute a HMAC and compare it to the signature.
3. If verification fails, respond with HTTP 401 and ignore the payload.
4. If verification succeeds, parse the task result and update the database.

5. Integration strategy

Because you already use a **port/adaptor pattern** and a **credit management system**, adding WaveSpeed is mostly an adaptor change. Follow these steps:

1. **Define a provider adapter** (SeedreamWaveSpeedAdapter) implementing your existing `ImageGenService` interface. It should:
2. Determine the correct endpoint based on `mode` (`text-to-image` → `/seedream-v4`; `image-edit` → `/seedream-v4-edit`).
3. Build the JSON body with your fields.
4. Send a POST request with `Authorization: Bearer <key>`.
5. Return the `taskId` to your domain layer.
6. Provide a method to check status via GET (if needed).
7. Normalize the provider's statuses (`succeeded`, `failed`, etc.) into your domain's `SUCCESS`, `FAILED`, etc.
8. Parse callback payloads after verifying the signature.
9. **Extend the callback route** so it can handle both NanoBanana and WaveSpeed. Use request headers (e.g., a signature header) to determine the provider. Validate accordingly and update tasks, credits, storage and refunds as you already do.

10. **Add configuration** variables in `.env`:

```
WAVESPEED_API_BASE=https://api.wavespeed.ai
WAVESPEED_API_KEY=your_wavespeed_api_key
PUBLIC_CALLBACK_URL=https://your-app.com/api/imagegen/callback
DEFAULT_IMAGE_PROVIDER=nanobanana  # or seedream for testing
```

1. **Update your credit system**:
2. Add a new provider record (`seedream`) to your `api_providers` table.
3. Define credit costs per task for Seedream. Seedream requests may use more credits because high-resolution images cost more to compute.
4. Leverage your existing credit deduction and refund functions (e.g., `consume_user_credits()` and `refund_user_credits()`) for Seedream as you do for NanoBanana【175†filecite】.
5. **Testing**:
6. Use development tunnels like ngrok or webhook.site to test callback delivery【177†filecite】.
7. Submit text-to-image and image-edit tasks; verify that results are returned and saved to Supabase.
8. Simulate failures to ensure credits are refunded and tasks are marked as failed.
9. Compare output quality and performance against NanoBanana to decide on default provider or weighting.

6. Sample usage flow

1. **User** clicks “Enhance” on a moodboard image.
2. **Frontend** sends POST `/api/enhance-image` with `mode`, `prompt`, `imageUrls` (if editing) and user token.
3. **API route** calls `ImageGenRouter.createTask()`, which delegates to either `NanoBananaAdapter` or `SeedreamWaveSpeedAdapter` based on environment or AB test configuration.
4. The **adapter** posts to WaveSpeed and stores the `taskId` and provider name in the `enhancement_tasks` table.
5. **Callback route** receives a POST from WaveSpeed. It verifies the signature, extracts the `taskId` and status, downloads the resulting images, saves them to Supabase and updates the task record. If failed, it refunds credits.
6. **Client** polls the `/api/enhance-image?taskId=` endpoint (or uses websockets) until the status changes to `completed` or `failed` and displays the resulting images or an error message.

7. Security considerations

- Use environment variables to store API keys.
- Always verify webhooks before processing them [2].
- Strip GPS and other sensitive EXIF data from uploaded and generated images (existing guidance applies).
- Monitor provider health and fallback to NanoBanana if WaveSpeed experiences errors or outages.

8. Conclusion

Bytedance Seedream V4 via WaveSpeed AI provides high-quality, multi-modal image generation and editing with fast inference and extensive editing features [1]. Thanks to your existing port/adaptor architecture and credit management system, you can add Seedream support with minimal changes: write a new adapter, verify webhooks, update your callback handler and adjust credit accounting. This dual-provider approach gives users a choice of AI engines and ensures redundancy if one provider fails.

---

[1] [2] Bytedance Seedream V4 API - Best Bytedance Seedream V4 API Pricing & Speed - WaveSpeedAI
https://wavespeed.ai/docs/docs-api/bytedance/bytedance-seedream-v4