ChatGPT

# NanoBanana API integration guide for Preset and Claude AI

## Overview of the NanoBanana API

The NanoBanana API is a cloud-hosted AI service designed to generate or edit images from natural-language descriptions or existing photos. It exposes REST endpoints for text-to-image generation and image editing, task status queries and credit management. Key features include:

- **Text-to-image generation** – submit a prompt describing the desired scene and the API returns one or more images. The API can generate up to four images per request [1].
- **Image editing** – supply a prompt and one or more input image URLs to transform existing images (e.g., alter colours or composition) [1].
- **Task management** – tasks run asynchronously. The API notifies your callback URL when a generation or editing task finishes [2]. You can also poll task status via the `Get Task Details` endpoint.
- **Credit management** – every call consumes credits. Use the credit API to check remaining credits and replenish as needed [1].

NanoBanana uses **Bearer-token authentication**. An API key must be created via the NanoBanana console and passed in the `Authorization` header as `Bearer YOUR_API_KEY` [2]. Keep your key secure and rotate it if compromised.

---

## Workflow summary

1. **Prepare a callback endpoint** – to receive task results, you must provide a publicly accessible HTTPS endpoint when submitting a generation or editing task. The API will send a POST request to this URL once the task completes. The callback must respond within 15 seconds. See the *Callback URL Best Practices* section below for recommendations [3].
2. **Submit a generation/editing request** (`POST /api/v1/nanobanana/generate`) – send a JSON body specifying the prompt, type (`TEXTTOIAMGE` for text-to-image or `IMAGETOIAMGE` for editing), callback URL and optional parameters (number of images, input image URLs and watermark). The request returns immediately with a `taskId` used to track progress.
3. **Handle the callback** – on success the callback payload includes the task ID and a `resultImageUrl` where your generated image is hosted. On failure it contains an error code and message. Implement idempotent processing since tasks may trigger repeated callbacks [3].
4. *(Optional)* **Poll for task status** (`GET /api/v1/nanobanana/record?taskId=`) – if your server cannot receive callbacks, poll this endpoint periodically (every ~30 s). The response returns generation status (`0` = generating, `1` = success, `2` = create task failed, `3` = generate failed) and, when complete, the original and result image URLs.

5. **Check credit balance** (`GET /api/v1/common/credit`) – monitor how many credits remain so that your application can warn users when running low [1].

---

## Endpoint reference

### 1. Submit an image generation or editing task

- **Method:** `POST`
- **Endpoint:** `/api/v1/nanobanana/generate`
- **Authentication:** Bearer token in the `Authorization` header [2]
- **Required body fields:** | Field | Type | Description | |------|------|-------------| | `prompt` | string | Natural-language description of the desired scene or the edit to apply. | | `type` | string | Either `TEXTTOIAMGE` (generate an image from text) or `IMAGETOIAMGE` (edit input images). | | `callBackUrl` | string (URI) | HTTPS webhook that will receive task completion notifications. |
- **Optional body fields:** | Field | Type | Description | |------|------|-------------| | `numImages` | integer | Number of images to generate (1–4). Default is 1 [3]. | | `imageUrls` | array of strings | For editing tasks, supply one or more publicly accessible image URLs. | | `watermark` | string | Optional watermark text applied to generated images. |

**Response:**

```
{
  "code": 200,
  "msg": "success",
  "data": {
    "taskId": "task12345"
  }
}
```

- `code` can be `200` (success), `400` (parameter error), `401` (unauthorized) or `500` (server error). - On success the `taskId` is used to track or poll task status.

**Important notes:** - You may generate **1–4 images** per request via the `numImages` parameter [3]. - A callback URL is **required** for receiving task completion notifications [3]. - Choose `TEXTTOIAMGE` for text-to-image generation or `IMAGETOIAMGE` for editing existing images [3]. - Task completion notifications are sent by HTTP POST to your callback URL; use the `Get Task Details` endpoint for polling.

### 2. Handle image generation/editing callbacks

Once a task finishes, the API sends a JSON payload to your `callBackUrl` with the following structure:

```
{
  "msg": "Image generated successfully.",
  "code": 200,
  "data": {
    "taskId": "9e4286b7b27960dfe8e1d279b50b28d",
    "info": {
      "resultImageUrl": "https://yourserver.com/image/result.jpg"
    }
  }
}
```

- `code` indicates task status: `200` success, `400` prompt flagged by content policy, `500` internal error or `501` generation failed [4].
- `data.taskId` matches the ID returned from the submission endpoint.
- `data.info.resultImageUrl` is the URL of the generated image on the NanoBanana server. Download it promptly because the original image URL (from the editing input) is valid for only 10 minutes [4].

**Callback URL best practices:**

- Use **HTTPS** and verify that requests come from trusted sources [5].
- Your callback handler should be **idempotent** because the same task may result in multiple callback attempts [5].
- Respond quickly with HTTP 200 to acknowledge receipt, then perform any heavy processing asynchronously [5].
- Download result images promptly after receiving the callback to avoid expiry [5].
- The server must respond within **15 seconds** or the API treats the callback as timed out [3]. After three retry failures, the API stops retrying.

If you cannot set up a callback, you may poll the `Get Task Details` endpoint instead [6].

## 3. Query task status

- **Method:** `GET`
- **Endpoint:** `/api/v1/nanobanana/record?taskId={taskId}`
- **Query parameter:** `taskId` (string) – the ID returned by the submission endpoint.
- **Authentication:** Bearer token.

**Response (success example):**

```
{
  "code": 200,
  "msg": "success",
  "data": {
    "taskId": "nanobanana_task_123456",
    "paramJson": "{…}",
    "completeTime": "2025-09-05T12:34:56Z",
    "response": {
      "originImageUrl": "https://bfl.com/image/original.jpg",
      "resultImageUrl": "https://ourserver.com/image/result.jpg"
    },
    "successFlag": 1,
    "errorCode": 0,
    "errorMessage": "",
    "createTime": "2025-09-05T12:33:00Z"
  }
}
```

- `successFlag` indicates task state: `0` generating, `1` success, `2` create task failed, `3` generation failed [7].
- `response.originImageUrl` is the original image (for editing tasks) and is valid for **10 minutes** [4].
- `response.resultImageUrl` is the generated image stored on NanoBanana's servers. It has longer availability and should be downloaded promptly.
- `errorCode` and `errorMessage` provide additional error information when `successFlag` is not 1 [7].

## 4. Get account credits

- **Method:** `GET`
- **Endpoint:** `/api/v1/common/credit`
- **Authentication:** Bearer token.

**Purpose:** Returns the number of remaining credits in your account. An example response is:

```
{
  "code": 200,
  "msg": "success",
  "data": 100
}
```

Possible response codes include: - `200` – success; - `401` – unauthorized (missing/invalid API key); - `402` – insufficient credits; the operation should not proceed [8]; - `404` – resource not found; - `422` – validation error (e.g. missing parameters); - `429` – rate limit exceeded; - `455` – service unavailable due to maintenance [8]; - `500` – server error; - `505` – feature disabled.

**Developer notes:** Credits are consumed based on the number of images generated and the service type. When credits run out, the API will refuse generation requests [9]. Monitor credit balance and top up accordingly.

---

## Integrating NanoBanana API with Preset

Preset will use NanoBanana to empower creators and talent with AI-assisted moodboard generation and image editing. Here's how to integrate it into the Preset backend:

1. **API key management:** Store the NanoBanana API key in a secure configuration service (e.g., environment variables in Vercel or Supabase secrets). Never expose it in client-side code.
2. **Server-side endpoint for generation:** Implement a serverless function (e.g., Vercel Function or Supabase Edge Function) that receives generation requests from the Preset app. Validate user input (prompt, optional editing images) and call the NanoBanana `POST /generate` endpoint with your API key.
3. **Callback handler:** Deploy a public webhook endpoint to handle NanoBanana callbacks. Verify the request source and parse the payload. When the result image URL is provided, download the image to your storage (e.g., Supabase Storage), associate it with the corresponding moodboard or Showcase entry, and update the Preset database with task status.
4. **Task polling fallback:** If a callback cannot be used (e.g., during local development), implement periodic polling using the `GET /record` endpoint until `successFlag = 1` or an error occurs. Avoid polling more frequently than every 30 seconds to respect rate limits [6].
5. **Credit awareness:** Before submitting a generation task, call `GET /credit` to check that the account has enough credits. Inform users when credit levels are low so they can upgrade or purchase more.
6. **Error handling:** Handle all possible error codes: adjust prompts when a `400` content-policy violation occurs; retry later for `500`/`501` server errors; stop processing when `402` (insufficient credits) is returned [4] [8].

---

## Using NanoBanana with Claude AI

To leverage the NanoBanana API from a conversational AI assistant like Claude AI, implement a custom tool that wraps the REST endpoints:

1. **Define a `GenerateImage` action** that accepts the user's prompt and optional settings. When invoked, it calls the Preset serverless function that forwards the request to NanoBanana.
2. **Provide asynchronous feedback:** After submitting the request, inform the user that the image is being generated and provide a link or a follow-up message once the callback indicates completion. The tool can poll the task status if necessary.
3. **Respect API usage and content policies:** Use prompt moderation filters to prevent prohibited content. Handle content-policy violations (status `400`) by asking the user to rephrase their description [4].
4. **Expose credit information:** Implement a `GetCredits` action so that the assistant can tell users how many generation credits remain [8].
5. **Secure handling:** All API calls should be performed server-side to protect the API key; never embed the key in client or conversation context.

With these guidelines, you can harness NanoBanana’s AI-powered image generation within the Preset platform and integrate it into an AI assistant like Claude to help users brainstorm and visualize creative shoots.

---

[1] [2] Welcome to NanoBanana API - Mint Starter Kit
https://docs.nanobananaapi.ai/

[3] [4] [5] [6] Image Generation or Editing Callbacks - Mint Starter Kit
https://docs.nanobananaapi.ai/nanobanana-api/generate-or-edit-image-callbacks

[7] Get Task Details - Mint Starter Kit
https://docs.nanobananaapi.ai/nanobanana-api/get-task-details

[8] [9] Get Account Credits - Mint Starter Kit
https://docs.nanobananaapi.ai/common-api/get-account-credits