DePIN.as API Developer Kit (v2026.3)

Welcome to the **DePIN.as Distributed Compute API**. This kit provides model keys, payload schemas, TypeScript definitions, and code examples for routing Text, Vision, and 3D workloads across our decentralized Apple Silicon node fleets.

1. Supported Model Identifiers & Fleet Endpoints

All workloads follow a two-step async lifecycle:

1. **Dispatch:** Submit workload payload via `POST /api/public/{type}/ingest` to receive a `job_id`.
2. **Poll Status:** Retrieve progress and output payload via `GET /api/public/{type}/status/{job_id}`.

Text Models

Dropdown Display Name	API Model Key (`"model"`)	Min. Node Fleet	Ingest Endpoint (`POST`)	Status Endpoint (`GET`)
Llama 3.2 3B (Standard)	`"llama-3.2-3b"`	**8GB+**	`/api/public/text/ingest`	`/api/public/text/status/{job_id}`
Qwen 3.7 Max MoE (Premium)	`"qwen-3.7-max"`	**32GB+**	`/api/public/text/ingest`	`/api/public/text/status/{job_id}`

Vision Models

Dropdown Display Name	API Model Key (`"model"`)	Min. Node Fleet	Ingest Endpoint (`POST`)	Status Endpoint (`GET`)
Qwen2.5-VL 3B (Standard)	`"qwen2.5-vl-3b"`	**8GB+**	`/api/public/vision/ingest`	`/api/public/vision/status/{job_id}`
Qwen3-VL 8B (Premium)	`"qwen3-vl-8b"`	**16GB+**	`/api/public/vision/ingest`	`/api/public/vision/status/{job_id}`
Qwen3-VL 32B (Premium)	`"qwen3-vl-32b"`	**32GB+**	`/api/public/vision/ingest`	`/api/public/vision/status/{job_id}`

3D Models (TRELLIS)

Dropdown Display Name	API Model Key (`"model"`)	Min. Node Fleet	Ingest Endpoint (`POST`)	Status Endpoint (`GET`)
TRELLIS 2.4B (Spatial Engine)	`"trellis-2.4b"`	**16GB+**	`/api/public/3d/ingest`	`/api/public/3d/status/{job_id}`

---

2. Standard JSON Payloads & Lifecycle Responses

Text Generation Payload (`sample_batch.json`)

```json
{
  "job_type": "text",
  "model": "qwen-3.7-max",
  "fallback_model": "llama-3.2-3b",
  "parameters": {
    "temperature": 0.7,
    "max_tokens": 2048
  },
  "prompts": [
    {
      "id": "req_001",
      "system_prompt": "You are an enterprise AI assistant.",
      "user_prompt": "Analyze the attached logistics dataset and summarize bottleneck risks."
    }
  ]
}
```

Step 1 Response (`POST /api/public/text/ingest`)

```json
{
  "ok": true,
  "job_id": "72ad9757-3c81-4b7d-8152-44169df301aa",
  "status": "Queued",
  "assigned_node_fleet": "32gb_plus"
}
```

Step 2 Response (`GET /api/public/text/status/72ad9757-3c81-4b7d-8152-44169df301aa`)

```json
{
  "ok": true,
  "job_id": "72ad9757-3c81-4b7d-8152-44169df301aa",
  "status": "Completed",
  "output_text": "Based on the logistics dataset provided...",
  "execution_time_ms": 1420,
  "error": null
}
```

Vision Payload (`sample_vision_batch.json`)

```json
{
  "job_type": "vision",
  "model": "qwen3-vl-32b",
  "fallback_model": "qwen2.5-vl-3b",
  "input": {
    "image_url": "https://assets.example.com/invoices/inv_9842.png",
    "prompt": "Extract all invoice line items, tax identification numbers, and totals as structured JSON."
  },
  "parameters": {
    "max_tokens": 1024
  }
}
```

3D Generation Payload (`sample_3d_batch.json`)

```json
{
  "job_type": "3d",
  "model": "trellis-2.4b",
  "execution_mode": "INFERENCE",
  "queue_target": "threed_jobs",
  "tasks": [
    {
      "id": "3d_task_001",
      "input_image_url": "https://assets.example.com/concept_asset.png",
      "output_format": "glb",
      "texture_resolution": 2048,
      "parameters": {
        "ss_sampling_steps": 12,
        "slat_sampling_steps": 12
      }
    }
  ]
}
```

—

3. TypeScript SDK Types (`depin-types.ts`)

```typescript
export type TextModelKey =
  | 'llama-3.2-3b'
  | 'qwen-3.7-max';

export type VisionModelKey =
  | 'qwen2.5-vl-3b'
  | 'qwen3-vl-8b'
  | 'qwen3-vl-32b';

export type ThreeDModelKey =
  | 'trellis-2.4b';

export type FleetTier = '8gb_plus' | '16gb_plus' | '32gb_plus';
export type ExecutionMode = 'INFERENCE' | 'BATCH';
export type JobStatus = 'Queued' | 'Processing' | 'Completed' | 'Failed';

export interface DePin3DTask {
  id: string;
  input_image_url: string;
  output_format: 'glb' | 'obj' | 'usdz';
  texture_resolution?: number;
  parameters?: {
    ss_sampling_steps?: number;
    slat_sampling_steps?: number;
  };
}

export interface DePin3DJobRequest {
  job_type: '3d';
  model: ThreeDModelKey;
  execution_mode: ExecutionMode;
  queue_target: 'threed_jobs';
  tasks: DePin3DTask[];
}

export interface DePinIngestResponse {
  ok: boolean;
  job_id: string;
  status: JobStatus;
  assigned_node_fleet?: FleetTier;
  error?: string;
}

export interface DePinJobResultResponse {
  ok: boolean;
  job_id: string;
  status: JobStatus;
  output_text?: string;
  result_urls?: string[];
  execution_time_ms?: number;
```

```
  error?: string;
  detail?: string;
}
```

---

4. Integration Code Examples

Python Async Submission & Polling (`3d_ingest.py`)

```python
import time
import requests

API_KEY = "depin_live_YOUR_SECRET_KEY"
BASE_URL = "https://api.depin.as/api/public/3d"

headers = {
    "Authorization": f"Bearer {API_KEY}",
    "Content-Type": "application/json"
}

payload = {
    "job_type": "3d",
    "model": "trellis-2.4b",
    "execution_mode": "INFERENCE",
    "queue_target": "threed_jobs",
    "tasks": [
        {
            "id": "chair_3d_01",
            "input_image_url": "https://assets.example.com/chair.png",
            "output_format": "glb"
        }
    ]
}

**1. Dispatch Job**
ingest_res = requests.post(f"{BASE_URL}/ingest", json=payload, headers=headers)
job_data = ingest_res.json()

if not job_data.get("ok"):
    raise Exception(f"Job Ingest Failed: {job_data.get('error')}")

job_id = job_data["job_id"]
print(f"Job Queued successfully. ID: {job_id}")

**2. Poll for Result**
status_url = f"{BASE_URL}/status/{job_id}"
timeout_seconds = 180
start_time = time.time()
```

```
while time.time() - start_time < timeout_seconds:
    status_res = requests.get(status_url, headers=headers).json()
    status = status_res.get("status")

    if status == "Completed":
        print("Job Completed!")
        print("Result Assets:", status_res.get("result_urls"))
        break
    elif status == "Failed":
        print("Job Execution Failed:", status_res.get("error"))
        break

    print(f"Status: {status}... checking again in 2s.")
    time.sleep(2)
```

---

cURL Async Lifecycle Example

Step 1: Dispatch 3D Job

```bash
curl -X POST https://api.depin.as/api/public/3d/ingest \
  -H "Authorization: Bearer depin_live_YOUR_SECRET_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "job_type": "3d",
    "model": "trellis-2.4b",
    "execution_mode": "INFERENCE",
    "queue_target": "threed_jobs",
    "tasks": [
      {
        "id": "asset_99",
        "input_image_url": "https://example.com/character_sheet.png",
        "output_format": "glb"
      }
    ]
  }'
```

Step 2: Poll Job Status & Fetch Result

```bash
curl -X GET https://api.depin.as/api/public/3d/status/
72ad9757-3c81-4b7d-8152-44169df301aa \
  -H "Authorization: Bearer depin_live_YOUR_SECRET_KEY"
```