GravixLayer File Management API Reference

Last Updated: September 11, 2025

Table of Contents

Authentication

All API requests require authentication using a Bearer token in the Authorization header:

```
Authorization: Bearer YOUR_API_KEY
```

The File Management API is OpenAI-compatible and uses the same authentication pattern as OpenAI's Files API.

File Management API

The File Management API provides OpenAI-compatible endpoints for uploading, managing, and retrieving files for use with AI models. All file operations are scoped to the authenticated user's account.

Base URL

```
/v1/files
```

File Object Structure

```json
{
  "id": "file-abc123def456",
  "object": "file",
  "bytes": 120000,
  "created_at": 1694808000,
  "filename": "dataset.jsonl",
  "purpose": "fine-tune",
  "expires_after": 86400
}
```

Fields: - `id` (string): Unique file identifier (UUID format) - `object` (string): Always "file" - `bytes` (integer): File size in bytes - `created_at` (integer): Unix timestamp of creation - `filename` (string): Original filename - `purpose` (string): File purpose/category - `expires_after` (integer, optional): Expiration time in seconds from creation

---

Upload File

POST `/v1/files`

Upload a file for use with AI models. Files are stored securely and associated with your account.

Request Format: `multipart/form-data`

Form Fields: - `file` (required): File to upload - **Max Size:** 200MB - **Min Size:** 1 byte - **Supported Types:** Any file type - `purpose` (required): File purpose - **Valid Values:** `assistants`, `batch`, `batch_output`, `fine-tune`, `vision`, `user_data`, `evals` - `expires_after` (optional): Expiration time in seconds - **Type:** Positive integer - **Description:** File will be automatically deleted after this time

Example Request:

```
curl -X POST https://api.gravixlayer.com/v1/files \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -F "file=@dataset.jsonl" \
  -F "purpose=fine-tune" \
  -F "expires_after=86400"
```

Success Response (200):

```
{
  "message": "file uploaded",
  "file_name": "dataset.jsonl",
  "purpose": "fine-tune"
}
```

Error Responses:

400 - Invalid Form Data:

```
{
  "error": "invalid form data"
}
```

400 - Invalid Purpose:

```
{
  "error": "invalid purpose. Supported: assistants, batch, batch_output,
fine-tune, vision, user_data, evals"
}
```

400 - Invalid File Size:

```
{
  "error": "file size must be between 1 byte and 200MB"
}
```

400 - Invalid Expiration:

```
{
  "error": "expires_after must be a positive integer (seconds)"
}
```

400 - Missing File:

```
{
  "error": "file is required"
}
```

409 - File Already Exists:

```
{
  "error": "file already exists"
}
```

List Files

GET `/v1/files`

Retrieve a list of all active files belonging to your account, ordered by creation date (newest first).

Query Parameters: None

Example Request:

```
curl -X GET https://api.gravixlayer.com/v1/files \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Success Response (200):

```
{
  "data": [
    {
      "id": "file-abc123def456",
      "object": "file",
      "bytes": 120000,
      "created_at": 1694808000,
      "filename": "dataset.jsonl",
      "purpose": "fine-tune",
      "expires_after": 86400
    },
    {
      "id": "file-def456ghi789",
      "object": "file",
      "bytes": 250000,
      "created_at": 1694807000,
      "filename": "training_data.txt",
      "purpose": "assistants"
    }
  ]
}
```

Empty Response:

```
{
  "data": []
}
```

Notes: - Expired files are automatically filtered out and marked as deleted - Only files with status "ready" are returned - Files are ordered by creation date (newest first)

Get File Metadata

GET `/v1/files/:id`

Retrieve metadata for a specific file by its ID.

Path Parameters: - `id` (required): File ID (UUID format)

Example Request:

```
curl -X GET https://api.gravixlayer.com/v1/files/file-abc123def456 \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Success Response (200):

```
{
  "id": "file-abc123def456",
  "object": "file",
  "bytes": 120000,
  "created_at": 1694808000,
  "filename": "dataset.jsonl",
  "purpose": "fine-tune",
  "expires_after": 86400
}
```

Error Responses:

400 - Missing File ID:

```
{
  "error": {
    "message": "file ID required"
  }
}
```

404 - File Not Found:

```
{
  "error": {
    "message": "file not found"
  }
}
```

404 - File Expired:

```
{
  "error": {
    "message": "file not found (expired)"
  }
}
```

Notes: - Files are automatically marked as deleted if they have expired - Only files belonging to your account are accessible

---

Get File Content

GET `/v1/files/:id/content`

Download the actual file content. This endpoint streams the file content directly.

Path Parameters: - `id` (required): File ID (UUID format)

Example Request:

```
curl -X GET https://api.gravixlayer.com/v1/files/file-abc123def456/content \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -o downloaded_file.jsonl
```

Success Response (200): - **Content-Type:** `application/octet-stream` - **Content-Disposition:** `attachment; filename=original_filename.ext` - **Body:** Raw file content (binary stream)

Error Responses:

400 - Missing File ID:

```
{
  "error": {
    "message": "file ID required"
  }
}
```

404 - File Not Found:

```
{
  "error": {
    "message": "file not found"
  }
}
```

500 - Storage Error:

```
{
  "error": {
    "message": "storage error"
  }
}
```

Notes: - Large files may take time to download (10-minute timeout) - File is streamed directly from storage - Original filename is preserved in Content-Disposition header

---

Delete File

DELETE `/v1/files/:id`

Delete a file permanently. This action cannot be undone.

Path Parameters: - `id` (required): File ID (UUID format)

Example Request:

```
curl -X DELETE https://api.gravixlayer.com/v1/files/file-abc123def456 \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Success Response (200):

```
{
  "message": "file deleted",
  "file_id": "file-abc123def456",
  "file_name": "dataset.jsonl"
}
```

Error Responses:

400 - Missing File ID:

```
{
  "error": "File ID is required"
}
```

404 - File Not Found:

```
{
  "error": "file not found"
}
```

500 - Delete Failed:

```json
{
  "error": "delete failed"
}
```

Notes: - File is removed from both database and storage - Only the file owner can delete files - Operation may take time for large files (10-minute timeout)

Error Handling

Error Response Format

File API errors follow a consistent format. Most errors use the nested format for OpenAI compatibility:

```json
{
  "error": {
    "message": "Error description"
  }
}
```

Some endpoints use the simplified format:

```json
{
  "error": "Error description"
}
```

Common HTTP Status Codes

- **200** - Success
- **400** - Bad Request (validation errors, missing parameters)
- **401** - Unauthorized (invalid or missing authentication)
- **404** - Not Found (file doesn’t exist or expired)
- **409** - Conflict (file already exists)
- **500** - Internal Server Error (storage or database errors)

File-Specific Error Scenarios

File Size Validation

- Files must be between 1 byte and 200MB
- Empty files are rejected
- Files over 200MB are rejected

Purpose Validation

Valid purposes: `assistants`, `batch`, `batch_output`, `fine-tune`, `vision`, `user_data`, `evals`

Expiration Handling

- Files with `expires_after` set are automatically deleted when expired

- Expired files return 404 errors
- Cleanup happens during list/get operations

File Ownership

- Users can only access their own files
- Files are isolated by user account and email
- Cross-account access is not permitted

Response Formats

File Object Fields

Field	Type	Description	Required
`id`	string	Unique file identifier (UUID)	Yes
`object`	string	Always “file”	Yes
`bytes`	integer	File size in bytes	Yes
`created_at`	integer	Unix timestamp of creation	Yes
`filename`	string	Original filename	Yes
`purpose`	string	File purpose/category	Yes
`expires_after`	integer	Expiration time in seconds	No

Success Response Types

Upload Response

```
{
  "message": "file uploaded",
  "file_name": "filename.ext",
  "purpose": "purpose_value"
}
```

List Response

```
{
  "data": [file_object, ...]
}
```

Metadata Response

```
file_object
```

Delete Response

```
{
  "message": "file deleted",
  "file_id": "file-id",
  "file_name": "filename.ext"
}
```

File Storage Details

Storage Architecture

- Files are stored in MinIO object storage
- Each user gets a dedicated bucket based on their email
- Bucket naming: `datasets-bucket-{email_prefix}`
- Email prefixes are sanitized (dots and underscores become dashes)

File Lifecycle

1. **Upload:** File is stored in MinIO and metadata in database
2. **Access:** Files are served directly from MinIO storage
3. **Expiration:** Automatic cleanup based on `expires_after` setting
4. **Deletion:** File removed from both storage and database

Security Features

- Files are isolated by user account
- Authentication required for all operations
- File ownership validation on all requests
- Automatic expiration and cleanup

Usage Examples

Complete File Upload Workflow

```
# 1. Upload a file
curl -X POST https://api.gravixlayer.com/v1/files \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -F "file=@training_data.jsonl" \
  -F "purpose=fine-tune" \
  -F "expires_after=604800"

# 2. List all files
curl -X GET https://api.gravixlayer.com/v1/files \
  -H "Authorization: Bearer YOUR_API_KEY"

# 3. Get file metadata
curl -X GET https://api.gravixlayer.com/v1/files/file-abc123 \
  -H "Authorization: Bearer YOUR_API_KEY"

# 4. Download file content
curl -X GET https://api.gravixlayer.com/v1/files/file-abc123/content \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -o downloaded_file.jsonl

# 5. Delete file
curl -X DELETE https://api.gravixlayer.com/v1/files/file-abc123 \
  -H "Authorization: Bearer YOUR_API_KEY"
```

JavaScript Example

```javascript
// Upload file
const formData = new FormData();
formData.append('file', fileInput.files[0]);
formData.append('purpose', 'assistants');
formData.append('expires_after', '86400');

const response = await fetch('/v1/files', {
  method: 'POST',
  headers: {
    'Authorization': 'Bearer YOUR_API_KEY'
  },
  body: formData
});

const result = await response.json();
console.log('File uploaded:', result);
```

Python Example

```python
import requests

# Upload file
files = {'file': open('dataset.jsonl', 'rb')}
data = {
    'purpose': 'fine-tune',
    'expires_after': '86400'
}
headers = {'Authorization': 'Bearer YOUR_API_KEY'}

response = requests.post(
    'https://api.gravixlayer.com/v1/files',
    files=files,
    data=data,
    headers=headers
)

print('Upload result:', response.json())
```

Notes and Limitations

1. **File Size Limits:** Maximum 200MB per file
2. **Storage Duration:** Files can be set to expire automatically
3. **Concurrency:** Multiple file operations can be performed simultaneously
4. **Timeout:** Upload/download operations have 10-minute timeouts
5. **File Types:** Any file type is supported
6. **Naming:** Original filenames are preserved
7. **Cleanup:** Expired files are automatically removed during operations

8. **Isolation:** Files are completely isolated between user accounts

This File Management API provides a robust, secure, and OpenAI-compatible interface for managing files in your AI applications.

Upload File

POST `/v1/files`

Upload a file for use with AI models (OpenAI-compatible).

Request Body: `multipart/form-data` - `file` (required): File to upload (max 200MB) - `purpose` (required): File purpose (`assistants`, `batch`, `batch_output`, `fine-tune`, `vision`, `user_data`, `evals`) - `expires_after` (optional): Expiration time in seconds

Response:

```
{
  "message": "file uploaded",
  "file_name": "dataset.jsonl",
  "purpose": "fine-tune"
}
```

List Files

GET `/v1/files`

Response:

```
{
  "data": [
    {
      "id": "file-abc123",
      "object": "file",
      "bytes": 120000,
      "created_at": 1694808000,
      "filename": "dataset.jsonl",
      "purpose": "fine-tune",
      "expires_after": 86400
    }
  ]
}
```

Get File Metadata

GET `/v1/files/:id`

Path Parameters: - `id` (required): File ID

Response:

```
{
  "id": "file-abc123",
  "object": "file",
  "bytes": 120000,
  "created_at": 1694808000,
  "filename": "dataset.jsonl",
  "purpose": "fine-tune",
  "expires_after": 86400
}
```

Get File Content

GET `/v1/files/:id/content`

Path Parameters: - `id` (required): File ID

Response: File content stream

Delete File

DELETE `/v1/files/:id`

Path Parameters: - `id` (required): File ID

Response:

```
{
  "message": "file deleted",
  "file_id": "file-abc123",
  "file_name": "dataset.jsonl"
}
```