PLATFORM DOCUMENTATION

CorePlexML

Release 0.1.0

Rodrigo Henriquez M.

Generated on Mar 09, 2026

Confidential - For internal use only
CorePlexML - Developed by Intellicore

ENDPOINTS

1 Base URL 3

2 Authentication 5

3 Common Response Format 7

4 Pagination 9

5 Content Types 11

6 Rate Limiting 13

7 File Upload Limits 15

8 Error Handling 17

9 PDF Download 19

This reference documents the CorePlexML REST API for developers building integrations, automating ML workflows, or consuming predictions. All endpoints accept and return JSON unless otherwise noted (file uploads use `multipart/form-data`).

CHAPTER

ONE

BASE URL

All paths in this documentation are relative to a configurable base URL. The default for local development is:

```
http://localhost:8888
```

In production the base URL is typically behind a reverse proxy with TLS, for example `https://ml.example.com`. All example `curl` commands below use the shell variable `$BASE_URL` which you should set to your own host:

```
export BASE_URL="http://localhost:8888"
```

CHAPTER

TWO

AUTHENTICATION

CorePlexML supports two authentication mechanisms:

Session cookies – Used by the web UI. Call `POST /api/auth/login` to obtain a session cookie that is automatically sent on subsequent requests.

Bearer tokens (API keys) – Recommended for programmatic access. Create an API key through `POST /api/auth/api-keys` and include it in every request via the `Authorization` header:

```
Authorization: Bearer YOUR_API_KEY
```

API keys carry scopes that restrict which operations they can perform. Available scopes: `read`, `write`, `predict`, `admin`.

> **Note**
>
> API key management endpoints (create/list/revoke) use cookie session auth and CSRF protection. See *Authentication* for the exact flow.

See *Authentication* for the full auth API.

CHAPTER

THREE

COMMON RESPONSE FORMAT

List endpoints return paginated results with a consistent envelope:

```
{
  "items": [ "..." ],
  "total": 142,
  "limit": 50,
  "offset": 0
}
```

Single-resource endpoints wrap the resource in a named key:

```
{
  "project": { "id": "...", "name": "..." }
}
```

Mutating endpoints that do not return a resource body return a confirmation:

```
{
  "ok": true
}
```

CHAPTER

FOUR

PAGINATION

Most list endpoints accept the following query parameters:

Parameter	Type	Default	Description
`limit`	integer	50	Maximum number of items to return (1–500).
`offset`	integer	0	Number of items to skip.
`search`	string	–	Free-text search (case-insensitive substring match).
`sort_field`	string	`created_`	Column to sort by. Allowed values vary per endpoint.
`sort_directi`	string	`desc`	`asc` or `desc`.

CHAPTER

FIVE

CONTENT TYPES

- **Request body** – `application/json` for all endpoints except file uploads.
- **File uploads** – `multipart/form-data` (datasets, batch predict).
- **Responses** – `application/json` unless the endpoint returns a binary file (CSV, PDF, XLSX, ZIP).

CHAPTER

SIX

RATE LIMITING

Endpoints are rate-limited per authenticated user using a fixed-window strategy. The server returns `X-RateLimit-Limit`, `X-RateLimit-Remaining`, and `X-RateLimit-Reset` headers on every response.

When the limit is exceeded, the server responds with **HTTP 429 Too Many Requests** and includes a `Retry-After` header.

Default rate limits (configurable by server administrator):

Endpoint Category	Limit	Description
General API	100/minute	Default for all endpoints not listed below.
Authentication	10/minute	Login, register, forgot-password. Prevents brute-force.
Predictions	50/minute	Model and deployment prediction endpoints.
AutoML Training	5/minute	Experiment creation (each starts a background job).
Report Generation	20/minute	Report creation (each starts a background job).
Dataset Builder (AI)	30/minute	AI-assisted data preparation endpoints.

CHAPTER

SEVEN

FILE UPLOAD LIMITS

Upload Type	Max Size	Description
Dataset CSV/Parquet	500 MB	Configurable via `MAX_DATASET_UPLOAD_BYTES` environment variable.
Batch prediction file	100 MB	CSV file for batch predictions.

Supported upload formats: `CSV` (.csv) and `Parquet` (.parquet).

CHAPTER

EIGHT

ERROR HANDLING

Errors are returned as JSON with an appropriate HTTP status code. The standard error body contains a `detail` field:

```
{
  "detail": "Project not found"
}
```

See *Error Handling* for the full list of status codes and common error scenarios.

CHAPTER

NINE

PDF DOWNLOAD

- API Reference PDF (EN): /pdfs/coreplexml-api-reference-en.pdf

9.1 Authentication

CorePlexML uses session cookies for web UI access and Bearer tokens (API keys) for programmatic access. Public endpoints that do **not** require authentication are: `POST /api/auth/register`, `POST /api/auth/login`, `POST /api/auth/forgot-password`, `POST /api/auth/reset-password`, and `GET /api/auth/verify-email`.

Important

API key management endpoints (`/api/auth/api-keys*`) require **cookie session authentication**. They do not accept Bearer API keys. For cookie-authenticated `POST/PUT/PATCH/DELETE` requests, include `X-CSRF-Token` with the value from the `csrf_token` cookie.

Endpoints

- *Register User*
- *Login*
- *Logout*
- *Get Current User*
- *Create API Key*
- *List API Keys*
- *Revoke API Key*
- *Forgot Password*
- *Reset Password*

9.1.1 Register User

```
POST /api/auth/register
```

Create a new user account.

Request Body

Field	Type	Required	Description
`email`	string	Yes	Valid email address.
`name`	string	Yes	Display name.
`password`	string	Yes	Minimum 8 characters.

Example

```
curl -X POST "$BASE_URL/api/auth/register" \
  -H "Content-Type: application/json" \
  -d '{
    "email": "alice@example.com",
    "name": "Alice Chen",
    "password": "s3cure!Pass"
  }'
```

```
import requests

resp = requests.post(f"{BASE_URL}/api/auth/register", json={
    "email": "alice@example.com",
    "name": "Alice Chen",
    "password": "s3cure!Pass",
})
print(resp.json())
```

Response `201 Created`

```
{
  "user": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "email": "alice@example.com",
    "name": "Alice Chen"
  },
  "verification_required": true
}
```

New accounts must verify email before they can log in.

9.1.2 Login

```
POST /api/auth/login
```

Authenticate and receive a session cookie.

Request Body

Field	Type	Required	Description
`email`	string	Yes	Registered email.
`password`	string	Yes	Account password.
`remember`	boolean	No	Extend session duration (default `false`).

Example

```
curl -X POST "$BASE_URL/api/auth/login" \
  -H "Content-Type: application/json" \
  -c cookies.txt \
  -d '{
    "email": "alice@example.com",
    "password": "s3cure!Pass"
  }'
```

```
session = requests.Session()
resp = session.post(f"{BASE_URL}/api/auth/login", json={
    "email": "alice@example.com",
    "password": "s3cure!Pass",
})
# session object now carries the cookie for subsequent requests
```

Response `200 OK`

```
{
  "user": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "email": "alice@example.com",
    "name": "Alice Chen"
  }
}
```

The response also sets an `HttpOnly` session cookie.

If the email is not verified, login returns `403` with detail: `Email not verified. Please check your inbox for the activation link.`.

9.1.3 Logout

```
POST /api/auth/logout
```

Revoke the current session.

Example

```
curl -X POST "$BASE_URL/api/auth/logout" \
  -b cookies.txt
```

Response `200 OK`

```
{
  "ok": true
}
```

9.1.4 Get Current User

```
GET /api/auth/me
```

Return profile information for the authenticated user.

Example

```
curl "$BASE_URL/api/auth/me" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(f"{BASE_URL}/api/auth/me", headers={
    "Authorization": "Bearer YOUR_API_KEY",
})
user = resp.json()
```

Response `200 OK`

```
{
  "user": {
    "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
    "email": "alice@example.com",
    "name": "Alice Chen",
    "is_superuser": false,
    "email_verified_at": "2026-01-15T10:30:00Z"
  }
}
```

9.1.5 Create API Key

```
POST /api/auth/api-keys
```

Generate a new API key for programmatic access. The secret is returned **only once** in the response – store it securely.

Request Body

Field	Type	Required	Description
`name`	string	Yes	Human-readable label for the key.
`scopes`	array[string]	No	Permissions: `read`, `write`, `predict`, `admin`. Default: all.

Example

```
# Requires a cookie-authenticated session and CSRF token
curl -X POST "$BASE_URL/api/auth/api-keys" \
  -b cookies.txt \
  -H "X-CSRF-Token: <csrf_token_cookie_value>" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "CI Pipeline Key",
    "scopes": ["*"]
  }'
```

```
csrf = session.cookies.get("csrf_token")
resp = session.post(f"{BASE_URL}/api/auth/api-keys", headers={
    "X-CSRF-Token": csrf,
}, json={
    "name": "CI Pipeline Key",
    "scopes": ["*"],
})
key_data = resp.json()
print("Save this key:", key_data["api_key"])
```

Response `201 Created`

```
{
  "prefix": "cp_a1b2c3d4",
  "api_key": "cp_a1b2c3d4.<secret>",
  "scopes": ["*"]
}
```

> **Warning**
>
> The `key` field is shown only in this response. If lost you must revoke the key and create a new one.

9.1.6 List API Keys

```
GET /api/auth/api-keys
```

Return all API keys for the current user. The secret portion of each key is **not** included.

Example

```
curl "$BASE_URL/api/auth/api-keys" \
  -b cookies.txt
```

Response `200 OK`

```
[
  {
    "id": "b2c3d4e5-f6a7-8901-bcde-f12345678901",
    "name": "CI Pipeline Key",
    "prefix": "cp_a1b2c3d4",
```

(continues on next page)

(continued from previous page)

```
    "scopes": ["*"],
    "last_used_at": "2026-02-28T18:45:00Z",
    "created_at": "2026-02-28T14:00:00Z"
  }
]
```

9.1.7 Revoke API Key

```
POST /api/auth/api-keys/{key_id}/revoke
```

Permanently revoke an API key. Subsequent requests using that key will return `401 Unauthorized`.

Path Parameters

Parameter	Type	Description
`key_id`	UUID	The ID of the API key to revoke.

Example

```
curl -X POST "$BASE_URL/api/auth/api-keys/b2c3d4e5-f6a7-8901-bcde-f12345678901/revoke" \
  -b cookies.txt \
  -H "X-CSRF-Token: <csrf_token_cookie_value>"
```

Response `200 OK`

```
{
  "ok": true
}
```

9.1.8 Forgot Password

```
POST /api/auth/forgot-password
```

Request a password reset email. Always returns `200` regardless of whether the email exists to prevent account enumeration.

Request Body

Field	Type	Required	Description
`email`	string	Yes	The email address associated with the account.

Example

```
curl -X POST "$BASE_URL/api/auth/forgot-password" \
  -H "Content-Type: application/json" \
  -d '{"email": "alice@example.com"}'
```

Response `200 OK`

```
{
  "ok": true
}
```

9.1.9 Reset Password

```
POST /api/auth/reset-password
```

Set a new password using a reset token received by email.

Request Body

Field	Type	Required	Description
`token`	string	Yes	Reset token from the email link (minimum 10 characters).
`password`	string	Yes	New password (minimum 8 characters).

Example

```
curl -X POST "$BASE_URL/api/auth/reset-password" \
  -H "Content-Type: application/json" \
  -d '{
    "token": "abc123def456ghi789jkl012mno345",
    "password": "newS3cure!Pass"
  }'
```

Response `200 OK`

```
{
  "ok": true
}
```

See also

- *Error Handling* – Error codes returned by auth endpoints (`401`, `403`, `429`).
- *Projects API* – Creating projects after authentication.

9.2 Projects API

Projects are the top-level organizational unit. Datasets, experiments, models, deployments, and all other resources belong to a project.

Endpoints

- *List Projects*
- *Create Project*
- *Get Project*
- *Update Project*
- *Delete Project*
- *List Members*
- *Add Member*
- *Activity Timeline*
- *Export Project*

9.2.1 List Projects

```
GET /api/projects
```

Return projects accessible to the authenticated user (owned or shared).

Query Parameters

Parameter	Type	Default	Description
`limit`	integer	50	Max items (1–500).
`offset`	integer	0	Pagination offset.
`search`	string	–	Search in name and description (case-insensitive).
`sort_field`	string	`created_`	One of `created_at`, `name`, `dataset_count`, `model_count`, `experiment_count`.
`sort_direction`	string	`desc`	`asc` or `desc`.

Example

```
curl "$BASE_URL/api/projects?limit=10&search=fraud" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
import requests

resp = requests.get(f"{BASE_URL}/api/projects", headers={
    "Authorization": "Bearer YOUR_API_KEY",
}, params={"limit": 10, "search": "fraud"})
data = resp.json()
for p in data["items"]:
    print(p["id"], p["name"])
```

Response `200 OK`

```
{
  "items": [
    {
```

(continues on next page)

(continued from previous page)

```
      "id": "c3d4e5f6-a7b8-9012-cdef-345678901234",
      "name": "Fraud Detection v2",
      "description": "Credit card fraud classification",
      "status": "active",
      "dataset_count": 3,
      "experiment_count": 5,
      "model_count": 12,
      "created_at": "2026-02-10T09:00:00Z"
    }
  ],
  "total": 1,
  "limit": 10,
  "offset": 0
}
```

9.2.2 Create Project

```
POST /api/projects
```

Request Body

Field	Type	Required	Description
`name`	string	Yes	Project name (must not be empty).
`description`	string	No	Optional description.

Example

```
curl -X POST "$BASE_URL/api/projects" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"name": "Churn Prediction", "description": "Telco customer churn"}'
```

```
resp = requests.post(f"{BASE_URL}/api/projects", headers={
    "Authorization": "Bearer YOUR_API_KEY",
}, json={
    "name": "Churn Prediction",
    "description": "Telco customer churn",
})
project = resp.json()
print("Project ID:", project["project_id"])
```

Response `201 Created`

```
{
  "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
  "project": {
    "id": "d4e5f6a7-b8c9-0123-def4-567890123456",
```

(continues on next page)

(continued from previous page)

```
    "name": "Churn Prediction",
    "description": "Telco customer churn",
    "created_at": "2026-02-28T14:30:00Z"
  }
}
```

9.2.3 Get Project

```
GET /api/projects/{project_id}
```

Path Parameters

Parameter	Type	Description
`project_id`	UUID	Project identifier.

Example

```
curl "$BASE_URL/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
print(resp.json()["project"]["name"])
```

Response `200 OK`

```
{
  "project": {
    "id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "name": "Churn Prediction",
    "description": "Telco customer churn",
    "created_at": "2026-02-28T14:30:00Z"
  }
}
```

9.2.4 Update Project

```
PUT /api/projects/{project_id}
```

Request Body

Field	Type	Required	Description
`name`	string	Yes	Updated name.
`description`	string	No	Updated description.

Example

```
curl -X PUT "$BASE_URL/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"name": "Churn Prediction v2", "description": "Updated scope"}'
```

```
resp = requests.put(
    f"{BASE_URL}/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    json={"name": "Churn Prediction v2", "description": "Updated scope"},
)
```

Response `200 OK`

```
{
  "project": {
    "id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "name": "Churn Prediction v2",
    "description": "Updated scope"
  }
}
```

9.2.5 Delete Project

```
DELETE /api/projects/{project_id}
```

Permanently delete a project. Only the project owner can perform this action.

Example

```
curl -X DELETE "$BASE_URL/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "ok": true
}
```

9.2.6 List Members

```
GET /api/projects/{project_id}/members
```

Return the list of users who have access to this project.

Query Parameters

Parameter	Type	Default	Description
`limit`	integer	50	Max items (1–100).
`offset`	integer	0	Pagination offset.

Example

```
curl "$BASE_URL/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456/members" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456/members",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
for member in resp.json()["items"]:
    print(member["email"], member["role"])
```

Response `200 OK`


```
{
  "items": [
    {
      "user_id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
      "email": "alice@example.com",
      "name": "Alice Chen",
      "role": "owner",
      "created_at": "2026-02-10T09:00:00Z"
    },
    {
      "user_id": "f6a7b8c9-0123-4567-89ab-cdef01234567",
      "email": "bob@example.com",
      "name": "Bob Smith",
      "role": "editor",
      "created_at": "2026-02-15T11:00:00Z"
    }
  ],
  "total": 2,
  "limit": 50,
  "offset": 0
}
```

9.2.7 Add Member

```
POST /api/projects/{project_id}/members
```

Add a user to the project or update their role. Only the project owner can manage members. You cannot assign a role higher than your own.

Request Body

Field	Type	Required	Description
`email`	string	Yes	Email of the user to add (must be a registered user).
`role`	string	No	One of `viewer`, `editor`, `admin`, `owner`. Default: `editor`.

Example

```
curl -X POST "$BASE_URL/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456/members" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"email": "bob@example.com", "role": "editor"}'
```

```
resp = requests.post(
    f"{BASE_URL}/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456/members",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    json={"email": "bob@example.com", "role": "editor"},
)
```

Response `201 Created`

```
{
  "member": {
    "user_id": "f6a7b8c9-0123-4567-89ab-cdef01234567",
    "email": "bob@example.com",
    "name": "Bob Smith",
    "role": "editor"
  }
}
```

9.2.8 Activity Timeline

```
GET /api/projects/{project_id}/timeline
```

Return a chronological list of events (datasets uploaded, experiments run) within the project. Useful for dashboards and audit trails.

Example

```
curl "$BASE_URL/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456/timeline" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
  "events": [
    {
      "id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
      "type": "dataset",
      "name": "transactions.csv",
      "timestamp": "2026-02-12T08:00:00Z",
      "rows": 50000,
      "columns": 15
    },
    {
      "id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
      "type": "experiment",
      "name": "AutoML Run 1",
```

(continues on next page)

(continued from previous page)

```
      "timestamp": "2026-02-12T10:30:00Z",
      "status": "completed",
      "target_column": "is_fraud",
      "problem_type": "classification",
      "best_model_type": "GBM",
      "duration": 342
    }
  ]
}
```

9.2.9 Export Project

```
GET /api/projects/{project_id}/export
```

Download the entire project as a ZIP archive containing a `manifest.json` and all associated artifacts (datasets, model binaries, reports).

Example

```
curl -o project_export.zip \
  "$BASE_URL/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456/export" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/projects/d4e5f6a7-b8c9-0123-def4-567890123456/export",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
with open("project_export.zip", "wb") as f:
    f.write(resp.content)
```

Response `200 OK`

Returns a binary ZIP file with `Content-Type: application/zip`.

See also

- *Datasets API* – Uploading data to a project.
- *Experiments API* – Running AutoML within a project.

9.3 Datasets API

Datasets are versioned tabular data files (CSV or Parquet) stored within a project. Each upload creates a dataset version with schema metadata and a data quality report.

Endpoints

- *Upload Dataset*
- *List Datasets*
- *Get Dataset Detail*
- *Get Column Schema*
- *Download Dataset*
- *Statistical Analysis*
- *Data Quality Report*
- *Dataset Lineage*
- *Delete Dataset*
- *List Versions*
- *Upload New Version*
- *Compare Versions*
- *Get Dataset Version*

9.3.1 Upload Dataset

```
POST /api/datasets/upload
```

Upload a CSV or Parquet file as a new dataset. Uses `multipart/form-data`. Maximum file size: 500 MB (configurable via `MAX_DATASET_UPLOAD_BYTES`).

Form Fields

Field	Type	Required	Description
`project_id`	string	Yes	UUID of the owning project.
`name`	string	Yes	Human-readable dataset name.
`description`	string	No	Optional description.
`file`	file	Yes	The CSV or Parquet file to upload.

Example

```
curl -X POST "$BASE_URL/api/datasets/upload" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -F "project_id=d4e5f6a7-b8c9-0123-def4-567890123456" \
  -F "name=Transactions Q4" \
  -F "description=Credit card transactions from Q4 2025" \
  -F "file=@transactions.csv"
```

```
import requests

with open("transactions.csv", "rb") as f:
    resp = requests.post(f"{BASE_URL}/api/datasets/upload", headers={
```

(continues on next page)

(continued from previous page)

```
        "Authorization": "Bearer YOUR_API_KEY",
    }, data={
        "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
        "name": "Transactions Q4",
        "description": "Credit card transactions from Q4 2025",
    }, files={"file": ("transactions.csv", f, "text/csv")})

data = resp.json()
print("Dataset ID:", data["dataset_id"])
print("Version ID:", data["dataset_version_id"])
```

Response `201 Created`

```
{
  "id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
  "dataset_id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
  "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
  "artifact_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
  "quality": {
    "overall_score": 92.5,
    "issues": []
  }
}
```

Supported File Formats

Format	Extension	Notes
CSV	`.csv`	Comma-separated values. Auto-detects delimiter, encoding, and header row.
Parquet	`.parquet`	Apache Parquet columnar format. Preserves column types.

Auto-Detected Column Types

The platform automatically detects column data types during upload:

Type	Description
`int64`	Integer values (whole numbers).
`float64`	Floating-point values (decimals).
`object`	String/text values, or mixed types.
`bool`	Boolean values (true/false).
`datetime64`	Date and time values (auto-parsed from common formats).
`category`	Categorical values (low-cardinality strings).

Download/Export Formats

Format	Description
CSV	Default export format for dataset downloads.
XLSX	Excel format (available for smaller datasets).

9.3.2 List Datasets

```
GET /api/datasets
```

Query Parameters

Parameter	Type	Default	Description
`project_id`	string	–	Filter by project. If omitted, returns datasets across all accessible projects.
`limit`	integer	50	Max items (1–500).
`offset`	integer	0	Pagination offset.
`search`	string	–	Search in name and description.
`sort_field`	string	`created_`	One of `created_at`, `name`, `rows`, `columns`, `project_name`.
`sort_direction`	string	`desc`	`asc` or `desc`.

Example

```bash
curl "$BASE_URL/api/datasets?project_id=d4e5f6a7-b8c9-0123-def4-567890123456&limit=20" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```python
resp = requests.get(f"{BASE_URL}/api/datasets", headers={
    "Authorization": "Bearer YOUR_API_KEY",
}, params={"project_id": "d4e5f6a7-b8c9-0123-def4-567890123456"})
for ds in resp.json()["items"]:
    print(ds["name"], ds["rows"], "rows")
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
      "name": "Transactions Q4",
      "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
      "project_name": "Fraud Detection v2",
      "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
      "rows": 50000,
      "columns": 15,
      "created_at": "2026-02-12T08:00:00Z"
    }
  ],
  "total": 1,
  "limit": 20,
  "offset": 0
}
```

9.3.3 Get Dataset Detail

```
GET /api/datasets/{dataset_id}
```

Return dataset metadata and a preview of the first 20 rows.

Example

```
curl "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "dataset": {
    "id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
    "name": "Transactions Q4",
    "rows": 50000,
    "columns": 15,
    "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
    "artifact_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456"
  },
  "preview": [
    {"amount": 29.99, "merchant": "ACME Corp", "is_fraud": 0},
    {"amount": 1250.00, "merchant": "Luxury Ltd", "is_fraud": 1}
  ],
  "preview_error": null
}
```

9.3.4 Get Column Schema

```
GET /api/datasets/{dataset_id}/columns
```

Return column names and data types for the latest dataset version.

Example

```
curl "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/columns" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "columns": [
    {"name": "amount", "dtype": "float64"},
    {"name": "merchant", "dtype": "object"},
    {"name": "is_fraud", "dtype": "int64"}
  ]
}
```

9.3.5 Download Dataset

```
GET /api/datasets/{dataset_id}/download
```

Download the latest version of a dataset as a file.

Query Parameters

Parameter	Type	Default	Description
`format`	string	`csv`	`csv` or `xlsx`.

Example

```
# Download as CSV
curl -o dataset.csv \
  "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/download" \
  -H "Authorization: Bearer YOUR_API_KEY"

# Download as Excel
curl -o dataset.xlsx \
  "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/download?format=xlsx" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

Binary file download.

9.3.6 Statistical Analysis

```
GET /api/datasets/{dataset_id}/analyze
```

Compute per-column statistics including mean, standard deviation, missing values, percentiles, skewness, kurtosis, outlier counts, histograms (numeric columns), and top values (categorical columns). Also returns a correlation matrix for numeric columns.

Example

```
curl "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/analyze" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/analyze",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
stats = resp.json()["statistics"]
for col, info in stats["columns"].items():
    print(f"{col}: type={info['type']}, missing={info['missing']}")
```

Response `200 OK`

```
{
  "statistics": {
    "columns": {
      "amount": {
        "type": "float64",
        "unique": 4520,
        "missing": 12,
        "missing_percent": 0.00024,
        "min": 0.01,
        "max": 25000.0,
        "mean": 88.35,
        "std": 340.12,
        "percentiles": {"p25": 9.99, "p50": 29.99, "p75": 69.99},
        "skewness": 15.234,
        "kurtosis": 312.5,
        "iqr": 60.0,
        "outlier_count": 482,
        "histogram": {
          "counts": [42000, 5000, 1500, 800, 300, 200, 100, 50, 30, 20],
          "bin_edges": [0.01, 2500.0, 5000.0, 7500.0, 10000.0, 12500.0, 15000.0, 17500.0,
↪ 20000.0, 22500.0, 25000.0]
        }
      },
      "merchant": {
        "type": "object",
        "unique": 120,
        "missing": 0,
        "missing_percent": 0.0,
        "top_values": {"ACME Corp": 5200, "Luxury Ltd": 3100}
      }
    },
    "row_count": 50000,
    "duplicate_rows": 45,
    "memory_usage": 5.8,
    "correlations": {
      "amount": {"amount": 1.0, "is_fraud": 0.15}
    }
  },
  "error": null
}
```


9.3.7 Data Quality Report

```
GET /api/datasets/{dataset_id}/quality
```

Return the data quality report computed during upload (stored in the dataset version).

Example

```
curl "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/quality" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "quality": {
    "overall_score": 92.5,
    "checks": [
      {"name": "missing_values", "passed": true, "details": "Max 0.02% missing"},
      {"name": "duplicate_rows", "passed": true, "details": "45 duplicates (0.09%)"}
    ]
  }
}
```

9.3.8 Dataset Lineage

```
GET /api/datasets/{dataset_id}/lineage
```

Trace the full lineage of a dataset: which experiments used it, which models were trained from those experiments, and which deployments serve those models.

Example

```
curl "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/lineage" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "experiments": [
    {
      "id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
      "name": "AutoML Run 1",
      "target_column": "is_fraud",
      "problem_type": "classification",
      "status": "succeeded",
      "model_count": 8
    }
  ],
  "models": [
    {
      "id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
      "name": "GBM_1_AutoML",
      "algorithm": "GBM",
      "experiment_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345"
    }
  ],
  "deployments": [
    {
      "id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
      "name": "GBM_1_AutoML",
      "model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
      "stage": "production",
      "is_active": true
```

(continues on next page)

(continued from previous page)

```
    }
  ]
}
```

9.3.9 Delete Dataset

```
DELETE /api/datasets/{dataset_id}
```

Permanently delete a dataset and all its versions and artifact files.

Example

```
curl -X DELETE "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "ok": true
}
```

9.3.10 List Versions

```
GET /api/datasets/{dataset_id}/versions
```

Return all versions of a dataset ordered by version number.

Query Parameters

Parameter	Type	Default	Description
`limit`	integer	50	Max items (1–100).
`offset`	integer	0	Pagination offset.

Example

```
curl "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/versions" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
      "dataset_id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
      "version": 0,
      "row_count": 50000,
      "column_count": 15,
```

(continues on next page)

(continued from previous page)


```
      "description": "Initial upload",
      "created_by_name": "Alice Chen",
      "created_at": "2026-02-12T08:00:00Z"
    },
    {
      "id": "a8b9c0d1-e2f3-4567-8901-abcdef234567",
      "dataset_id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
      "version": 1,
      "row_count": 75000,
      "column_count": 15,
      "description": "Added January transactions",
      "created_by_name": "Alice Chen",
      "created_at": "2026-02-20T11:00:00Z"
    }
  ],
  "total": 2,
  "limit": 50,
  "offset": 0
}
```


9.3.11 Upload New Version

```
POST /api/datasets/{dataset_id}/versions
```

Upload a new version of an existing dataset. Uses `multipart/form-data`.

Form Fields

Field	Type	Required	Description
`description`	string	Yes	Description of what changed in this version.
`file`	file	Yes	The CSV or Parquet file.

Example

```
curl -X POST "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/versions" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -F "description=Added January transactions" \
  -F "file=@transactions_v2.csv"
```

Response `201 Created`


```
{
  "dataset_version_id": "a8b9c0d1-e2f3-4567-8901-abcdef234567",
  "version": 1,
  "row_count": 75000,
  "column_count": 15
}
```

9.3.12 Compare Versions

```
GET /api/datasets/{dataset_id}/compare?v1=0&v2=1
```

Compare two versions of a dataset. Returns schema diffs (added/removed columns, type changes) and statistical drift metrics (PSI for numeric columns, category changes for categorical columns).

Query Parameters

Parameter	Type	Required	Description
v1	integer	Yes	First version number.
v2	integer	Yes	Second version number.

Example

```
curl "$BASE_URL/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/compare?v1=0&v2=1" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/datasets/e5f6a7b8-c9d0-1234-5678-90abcdef1234/compare",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    params={"v1": 0, "v2": 1},
)
summary = resp.json()["summary"]
print(f"Row change: {summary['rows_diff']:+d}")
```

Response `200 OK`


```
{
  "summary": {
    "v1": 0,
    "v2": 1,
    "v1_rows": 50000,
    "v2_rows": 75000,
    "rows_diff": 25000,
    "v1_columns": 15,
    "v2_columns": 15,
    "columns_added": [],
    "columns_removed": [],
    "columns_type_changed": []
  },
  "column_diffs": {
    "amount": {
      "type": "numeric",
      "v1_mean": 88.35,
      "v2_mean": 92.10,
      "v1_std": 340.12,
      "v2_std": 355.40,
      "missing_v1": 0.0002,
      "missing_v2": 0.0001,
      "mean_shift_pct": 4.25,
      "psi": 0.023,
```


(continues on next page)

(continued from previous page)

```
      "shift_level": "low"
    }
  }
}
```

PSI (Population Stability Index) interpretation:

- `< 0.1` – **low** drift, distributions are similar.
- `0.1 -- 0.25` – **moderate** drift, investigation recommended.
- `> 0.25` – **high** drift, significant distribution change.

9.3.13 Get Dataset Version

```
GET /api/dataset-versions/{dataset_version_id}
```

Return metadata for a specific dataset version including its column schema.

Example

```
curl "$BASE_URL/api/dataset-versions/f6a7b8c9-d0e1-2345-6789-0abcdef12345" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
  "dataset_id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
  "version": 0,
  "row_count": 50000,
  "column_count": 15,
  "columns": [
    {"name": "amount", "dtype": "float64"},
    {"name": "merchant", "dtype": "object"},
    {"name": "is_fraud", "dtype": "int64"}
  ],
  "created_at": "2026-02-12T08:00:00Z"
}
```

See also

- *Experiments API* – Training models on a dataset.
- *Privacy Suite API* – Scanning datasets for PII.

9.4 Experiments API

Experiments run AutoML training on a dataset. Creating an experiment enqueues a background job that trains multiple models and selects the best one according to the configured metric.

Endpoints

- *Create Experiment*
- *List Experiments*
- *Get Experiment Detail*
- *Get Experiment Status*
- *Get Training Logs*
- *Explainability*
- *Set Best Model*
- *Delete Experiment*

9.4.1 Create Experiment

```
POST /api/experiments
```

Create a new experiment and enqueue an `automl_train` background job. The experiment starts in `queued` status and transitions through `running` to `succeeded` or `failed`.

Request Body

Field	Type	Required	Description
`project_id`	string	Yes	UUID of the project.
`dataset_version_i`	string	Yes	UUID of the dataset version to train on.
`name`	string	Yes	Human-readable experiment name.
`target_column`	string	Yes	Name of the column to predict.
`problem_type`	string	Yes	`classification` or `regression`. Also accepts `binary` / `multiclass` (mapped to `classification`).
`config`	object	No	AutoML configuration (see below).
`use_gpu`	boolean	No	Request GPU-enabled training (default `false`).

Config Options

Key	Type	Description
`max_runtime_secs`	integer	Maximum training time in seconds (default 3600).
`max_models`	integer	Maximum number of models to train (default 20). May be capped by account quota.
`max_runtime_secs_per_`	integer	Maximum training time per individual model (optional).
`sort_metric`	string	Metric to rank models by (see *Sort & Stopping Metrics* below). Default `AUTO` (AUC for classification, RMSE for regression).
`seed`	integer	Random seed for reproducibility (default 42).
`balance_classes`	boolean	Oversample minority class (classification only, default `false`).
`exclude_algos`	array[string]	Algorithms to exclude (see *Supported Algorithms* below).
`nfolds`	integer	Number of cross-validation folds (default 5, use `-1` for auto-detection, `0` to disable).
`stopping_rounds`	integer	Number of scoring rounds without improvement before early stopping (default 3, use `0` to disable).
`stopping_metric`	string	Metric to monitor for early stopping: `AUTO`, `deviance`, `logloss`, `RMSE`, `AUC`.
`stopping_tolerance`	float	Relative tolerance for early stopping (default 0.001).
`exploitation_ratio`	float	Ratio of time for fine-tuning vs. exploring new models (0.0–1.0).
`distribution`	string	Distribution family (regression): `gaussian`, `poisson`, `gamma`, `tweedie`, `huber`, `quantile`, `laplace`.
`keep_cross_validation_`	boolean	Keep CV predictions for Stacked Ensembles (default `true`).
`keep_cross_validation_`	boolean	Keep individual CV models (default `false`).
`verbosity`	string	Logging level: `debug`, `info`, `warn`.

Supported Algorithms

CorePlexML uses H2O AutoML which automatically trains, tunes, and ranks models from the following algorithm families:

Algorithm ID	Display Name	Description
`GBM`	Gradient Boosting	Gradient Boosting Machine. Builds sequential decision trees where each tree corrects errors of the previous ones. Strong default performance on most tabular datasets.
`XGBoost`	XGBoost	Extreme Gradient Boosting. Optimized implementation with regularization options (L1/L2). Often produces top-performing models, especially on structured data.
`DRF`	Random Forest	Distributed Random Forest. Ensemble of decision trees using random feature subsets and bootstrap sampling. Robust to overfitting.
`DeepLearning`	Deep Learning	Multi-layer feed-forward neural network. Multiple hidden layers with configurable activation functions. Best for datasets with complex nonlinear relationships.
`GLM`	Linear Model	Generalized Linear Model. Includes linear regression, logistic regression, and other link functions. Fast, interpretable, and useful as a baseline.
`StackedEnsemble`	Stacked Ensemble	Meta-model that combines predictions from all other trained models using a secondary learner. Typically achieves the best accuracy. Note: SHAP feature contributions are **not available** for StackedEnsemble models.

All six algorithms are trained by default. Use `exclude_algos` to skip specific ones:

```
{
  "config": {
    "exclude_algos": ["DeepLearning", "StackedEnsemble"]
  }
}
```

Sort & Stopping Metrics

The `sort_metric` determines how models are ranked on the leaderboard. The `stopping_metric` determines when early stopping triggers. Set to `AUTO` (default) and the platform selects the best metric for your problem type.

Binary Classification Metrics

Metric	Description
`AUC`	Area Under ROC Curve (default for binary classification). Higher is better. Range: 0–1.
`AUCPR`	Area Under Precision-Recall Curve. Better than AUC for imbalanced datasets.
`logloss`	Logarithmic loss (cross-entropy). Lower is better.
`mean_per_class_error`	Average error rate across classes. Lower is better.
`accuracy`	Classification accuracy. Higher is better. Not recommended for imbalanced classes.
`MCC`	Matthews Correlation Coefficient. Balanced metric for imbalanced data. Range: -1 to 1.

Multiclass Classification Metrics

Metric	Description
`logloss`	Multinomial log loss (default for multiclass). Lower is better.
`mean_per_class_error`	Average error rate across all classes. Lower is better.

Regression Metrics

Metric	Description
`RMSE`	Root Mean Squared Error (default for regression). Lower is better.
`MSE`	Mean Squared Error. Lower is better.
`MAE`	Mean Absolute Error. Less sensitive to outliers. Lower is better.
`RMSLE`	Root Mean Squared Logarithmic Error. Good for skewed targets. Lower is better.
`mean_residual_devianc`	Mean Residual Deviance. Lower is better.
`R2`	R-squared (coefficient of determination). Higher is better. Range: 0–1.

Distribution-Specific Metrics

Metric	Description
`quantile_loss`	Quantile loss. Available when `distribution` is `quantile` or `huber`.
`deviance`	Deviance. Available for all distributions. Lower is better.

Problem Types

Value	Description
`classification`	Binary or multiclass classification. Auto-detected from the target column cardinality.
`regression`	Continuous numeric prediction.
`binary`	Alias for `classification` (mapped automatically).
`multiclass`	Alias for `classification` (mapped automatically).

Experiment Status Lifecycle

An experiment transitions through these statuses:

Status	Description
`queued`	Experiment created and waiting for worker to pick it up.
`running`	AutoML training in progress. Models are being trained.
`completed` / `succeeded`	Training finished successfully. Models are available on the leaderboard. (API returns `succeeded`, UI displays `completed`.)
`failed`	Training failed due to an error (bad data, configuration issue, or system error). Check experiment logs for details.

Job Status Values

Background jobs (experiments, reports, synthgen) use these statuses:

Status	Description
`queued`	Job created, waiting for worker.
`running`	Worker is processing the job.
`succeeded`	Job completed successfully.
`failed`	Job failed. Error details in job payload.

Example

```
curl -X POST "$BASE_URL/api/experiments" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
    "name": "Fraud Detection AutoML",
    "target_column": "is_fraud",
    "problem_type": "classification",
    "config": {
      "max_runtime_secs": 300,
      "max_models": 20,
      "sort_metric": "auc",
      "nfolds": 5,
      "balance_classes": true,
      "exclude_algos": ["DeepLearning"]
    }
  }'
```

```python
import requests
import time

# Create the experiment
resp = requests.post(f"{BASE_URL}/api/experiments", headers={
    "Authorization": "Bearer YOUR_API_KEY",
}, json={
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
    "name": "Fraud Detection AutoML",
    "target_column": "is_fraud",
    "problem_type": "classification",
    "config": {
        "max_runtime_secs": 300,
        "max_models": 20,
        "sort_metric": "auc",
        "nfolds": 5,
        "balance_classes": True,
    },
})
experiment_id = resp.json()["experiment_id"]

# Poll for completion
while True:
    status_resp = requests.get(
        f"{BASE_URL}/api/experiments/{experiment_id}/status",
        headers={"Authorization": "Bearer YOUR_API_KEY"},
    )
    status = status_resp.json()["status"]
    print(f"Status: {status}")
    if status in ("succeeded", "failed"):
        break
    time.sleep(10)
```

Response `201 Created`

```json
{
  "id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
  "experiment_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
  "job_id": "c9d0e1f2-a3b4-5678-9012-cdef12345678",
  "status": "queued"
}
```

9.4.2 List Experiments

```
GET /api/experiments
```

Query Parameters

Parameter	Type	Default	Description
`project_id`	string	–	Filter by project.
`dataset_id`	string	–	Filter by source dataset.
`status`	string	–	Filter by status: `queued`, `running`, `succeeded`, `failed`.
`limit`	integer	50	Max items (1–500).
`offset`	integer	0	Pagination offset.
`search`	string	–	Search in name and target column.
`sort_field`	string	`created_`	`created_at`, `name`, or `status`.
`sort_direction`	string	`desc`	`asc` or `desc`.

Example

```
curl "$BASE_URL/api/experiments?project_id=d4e5f6a7-b8c9-0123-def4-567890123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
      "name": "Fraud Detection AutoML",
      "project_name": "Fraud Detection v2",
      "dataset_name": "Transactions Q4",
      "target_column": "is_fraud",
      "problem_type": "classification",
      "status": "succeeded",
      "model_count": 12,
      "best_model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
      "best_model_algo": "GBM",
      "created_at": "2026-02-12T10:30:00Z"
    }
  ],
  "total": 1,
  "limit": 50,
  "offset": 0
}
```

9.4.3 Get Experiment Detail

```
GET /api/experiments/{experiment_id}
```

Return the experiment, its models, artifact links, and the best model ID.

Example

```
curl "$BASE_URL/api/experiments/b8c9d0e1-f2a3-4567-8901-bcdef1234567" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/experiments/b8c9d0e1-f2a3-4567-8901-bcdef1234567",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
data = resp.json()
print("Best model:", data["best_model_id"])
for m in data["models"]:
    print(f"  {m['algo']}: {m['metrics'].get('auc', 'N/A')}")
```

Response `200 OK`


```
{
  "experiment": {
    "id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
    "name": "Fraud Detection AutoML",
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
    "target_column": "is_fraud",
    "problem_type": "classification",
    "status": "succeeded",
    "max_runtime_secs": 300,
    "max_models": 20,
    "sort_metric": "auc"
  },
  "models": [
    {
      "id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
      "name": "GBM_1_AutoML",
      "algo": "GBM",
      "metrics": {"auc": 0.982, "logloss": 0.071}
    }
  ],
  "artifact_links": [],
  "best_model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456"
}
```

9.4.4 Get Experiment Status

```
GET /api/experiments/{experiment_id}/status
```

Lightweight polling endpoint that returns only the current status. Use this for progress polling instead of the full detail endpoint.

Example

```
curl "$BASE_URL/api/experiments/b8c9d0e1-f2a3-4567-8901-bcdef1234567/status" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`


```
{
  "experiment_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
```


(continues on next page)

(continued from previous page)

```
  "status": "running"
}
```

Possible status values: `queued`, `running`, `succeeded`, `failed`, `canceled`.

9.4.5 Get Training Logs

```
GET /api/experiments/{experiment_id}/logs
```

Return training logs for debugging and monitoring. Includes H2O engine output, job status, and error messages.

Query Parameters

Parameter	Type	Default	Description
`include_h2o_logs`	boolean	`true`	Include detailed H2O event logs.

Example

```
curl "$BASE_URL/api/experiments/b8c9d0e1-f2a3-4567-8901-bcdef1234567/logs" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "experiment_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
  "logs": [
    "Experiment status: succeeded",
    "Job status: completed",
    "",
    "AutoML progress: 100%",
    "Best model: GBM_1_AutoML (AUC=0.982)"
  ],
  "count": 5
}
```

9.4.6 Explainability

```
GET /api/experiments/{experiment_id}/explain
```

Return multi-model explainability data including a Pareto front (accuracy vs. training time) and a variable importance heatmap across all models in the experiment.

Example

```
curl "$BASE_URL/api/experiments/b8c9d0e1-f2a3-4567-8901-bcdef1234567/explain" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "explain": {
    "pareto_front": {
      "models": [
        {"name": "GBM_1_AutoML", "algo": "GBM", "metric": 0.982, "training_time": 45.2},
        {"name": "XGBoost_1_AutoML", "algo": "XGBoost", "metric": 0.978, "training_time
↪": 32.1}
      ]
    },
    "varimp_heatmap": {
      "models": ["GBM_1_AutoML", "XGBoost_1_AutoML"],
      "features": ["amount", "merchant_id", "hour"],
      "matrix": [
        [0.45, 0.30, 0.15],
        [0.42, 0.35, 0.12]
      ]
    }
  }
}
```

9.4.7 Set Best Model

```
POST /api/experiments/{experiment_id}/best-model
```

Override the default best model selection for an experiment.

Request Body

Field	Type	Required	Description
`model_id`	string	Yes	UUID of a model that belongs to this experiment.

Example

```
curl -X POST "$BASE_URL/api/experiments/b8c9d0e1-f2a3-4567-8901-bcdef1234567/best-model"␣
↪\
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456"}'
```

Response `200 OK`

```
{
  "ok": true,
  "experiment_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
  "best_model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456"
}
```

9.4.8 Delete Experiment

```
DELETE /api/experiments/{experiment_id}
```

Delete an experiment and its associated models.

Example

```
curl -X DELETE "$BASE_URL/api/experiments/b8c9d0e1-f2a3-4567-8901-bcdef1234567" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "ok": true
}
```

See also

- *Models API* – Inspecting individual models from an experiment.
- *Deployments API* – Deploying the best model to production.

9.5 Models API

Models are machine learning models produced by AutoML experiments. Each model stores its algorithm type, evaluation metrics, and a binary artifact that can be used for predictions.

Endpoints

- *List Models*
- *Get Model Detail*
- *Get Hyperparameters*
- *Model Explainability*
- *Single/Batch Prediction*
- *Batch Predict from File*
- *Delete Model*

9.5.1 List Models

```
GET /api/models
```

Query Parameters

Parameter	Type	Default	Description
`project_id`	string	–	Filter by project.
`experiment_id`	string	–	Filter by experiment.
`model_type`	string	–	Filter by algorithm (e.g. `GBM`, `XGBoost`, `GLM`).
`is_best`	boolean	–	Only return the best model per experiment.
`is_production`	boolean	–	Only return models with an active production deployment.
`is_staging`	boolean	–	Only return models with an active staging deployment.
`search`	string	–	Search in model name, algorithm, experiment name, project name.
`limit`	integer	50	Max items.
`offset`	integer	0	Pagination offset.
`sort_metric`	string	–	Sort by a metric value: `auc`, `rmse`, `mae`, `r2`, `logloss`, `accuracy`, `mse`, `mean_per_class_error`.
`sort_dir`	string	`desc`	`asc` or `desc`.

Example

```
curl "$BASE_URL/api/models?project_id=d4e5f6a7-b8c9-0123-def4-567890123456&sort_
↪metric=auc" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
import requests

resp = requests.get(f"{BASE_URL}/api/models", headers={
    "Authorization": "Bearer YOUR_API_KEY",
}, params={
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "sort_metric": "auc",
    "sort_dir": "desc",
    "limit": 5,
})
for m in resp.json()["items"]:
    print(f"{m['algo']}: AUC={m['metrics'].get('auc')}")
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
      "name": "GBM_1_AutoML",
      "algo": "GBM",
      "experiment_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
      "experiment_name": "Fraud Detection AutoML",
      "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
      "project_name": "Fraud Detection v2",
      "is_best": true,
      "is_production": true,
      "is_staging": false,
      "metrics": {
        "auc": 0.982,
        "logloss": 0.071,
        "accuracy": 0.965
```

(continues on next page)

(continued from previous page)

```
      },
      "created_at": "2026-02-12T10:35:00Z"
    }
  ],
  "total": 12,
  "limit": 5,
  "offset": 0
}
```

9.5.2 Get Model Detail

```
GET /api/models/{model_id}
```

Return model metadata, evaluation metrics, and artifact links.

Example

```
curl "$BASE_URL/api/models/a7b8c9d0-e1f2-3456-7890-abcdef123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "model": {
    "id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
    "name": "GBM_1_AutoML",
    "algo": "GBM",
    "experiment_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
    "artifact_id": "c9d0e1f2-a3b4-5678-9012-cdef23456789",
    "metrics": {
      "auc": 0.982,
      "logloss": 0.071,
      "accuracy": 0.965,
      "confusion_matrix": {
        "labels": ["0", "1"],
        "matrix": [[48500, 200], [150, 1150]]
      },
      "variable_importance": [
        {"variable": "amount", "percentage": 0.45},
        {"variable": "merchant_id", "percentage": 0.30}
      ]
    }
  },
  "artifact_links": [
    {
      "artifact_id": "c9d0e1f2-a3b4-5678-9012-cdef23456789",
      "owner_type": "model",
      "label": "model_binary"
    }
  ]
}
```

9.5.3 Get Hyperparameters

```
GET /api/models/{model_id}/parameters
```

Return the hyperparameters used to train the model. If not stored in metadata, they are loaded from the H2O model artifact.

Example

```
curl "$BASE_URL/api/models/a7b8c9d0-e1f2-3456-7890-abcdef123456/parameters" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "parameters": {
    "ntrees": {"default": 50, "actual": 150},
    "max_depth": {"default": 5, "actual": 8},
    "learn_rate": {"default": 0.1, "actual": 0.05},
    "sample_rate": {"default": 1.0, "actual": 0.8}
  }
}
```

9.5.4 Model Explainability

```
GET /api/models/{model_id}/explain
```

Return structured explainability data for a single model: variable importance, SHAP values, confusion matrix, ROC curve, precision-recall curve, scoring history, and more.

Example

```
curl "$BASE_URL/api/models/a7b8c9d0-e1f2-3456-7890-abcdef123456/explain" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "explain": {
    "model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
    "algo": "GBM",
    "variable_importance": [
      {"variable": "amount", "percentage": 0.45, "scaled_importance": 1.0},
      {"variable": "merchant_id", "percentage": 0.30, "scaled_importance": 0.67}
    ],
    "shap_summary": null,
    "confusion_matrix": {
      "labels": ["0", "1"],
      "matrix": [[48500, 200], [150, 1150]]
    },
```

(continues on next page)

(continued from previous page)

```
    "roc_curve": {
      "fpr": [0.0, 0.004, 0.01, 1.0],
      "tpr": [0.0, 0.85, 0.95, 1.0]
    },
    "scoring_history": [
      {"timestamp": "2026-02-12T10:31:00Z", "training_logloss": 0.35, "validation_logloss
↪": 0.38}
    ]
  }
}
```

9.5.5 Single/Batch Prediction

```
POST /api/models/{model_id}/predict
```

Run predictions directly on a model (without a deployment). Accepts a single input dict or a list of input dicts.

Request Body

Field	Type	Required	Description
`inputs`	dict or list[dict]	Yes	Feature values. Single record (dict) or batch (list of dicts).
`options`	object	No	Options. Set `include_contributions: true` for SHAP-style feature contributions (not available for StackedEnsemble models).

Example

```
curl -X POST "$BASE_URL/api/models/a7b8c9d0-e1f2-3456-7890-abcdef123456/predict" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "inputs": [
      {"amount": 29.99, "merchant_id": 42, "hour": 14},
      {"amount": 9500.00, "merchant_id": 7, "hour": 3}
    ],
    "options": {"include_contributions": true}
  }'
```

```
resp = requests.post(
    f"{BASE_URL}/api/models/a7b8c9d0-e1f2-3456-7890-abcdef123456/predict",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    json={
        "inputs": [
            {"amount": 29.99, "merchant_id": 42, "hour": 14},
            {"amount": 9500.00, "merchant_id": 7, "hour": 3},
        ],
        "options": {"include_contributions": True},
    },
```

(continues on next page)

(continued from previous page)

```
)
for pred in resp.json()["predictions"]:
    print("Prediction:", pred["prediction"])
```

Response `200 OK`

```
{
  "predictions": [
    {"prediction": "0"},
    {"prediction": "1"}
  ],
  "contributions": [
    [
      {"feature": "amount", "value": -0.32},
      {"feature": "merchant_id", "value": -0.15},
      {"feature": "BiasTerm", "value": -1.20}
    ],
    [
      {"feature": "amount", "value": 1.85},
      {"feature": "hour", "value": 0.92},
      {"feature": "BiasTerm", "value": -1.20}
    ]
  ]
}
```

9.5.6 Batch Predict from File

```
POST /api/models/{model_id}/predict/batch
```

Upload a CSV or Parquet file and receive predictions for all rows. Uses `multipart/form-data`.

Form Fields

Field	Type	Required	Description
`file`	file	Yes	CSV or Parquet file with input features.

Example

```
curl -X POST "$BASE_URL/api/models/a7b8c9d0-e1f2-3456-7890-abcdef123456/predict/batch" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -F "file=@new_transactions.csv"
```

Response `200 OK`

```
{
  "row_count": 1000,
  "predictions": [
    {"amount": 29.99, "merchant_id": 42, "pred_predict": "0"},
```

(continues on next page)

(continued from previous page)

```
    {"amount": 9500.00, "merchant_id": 7, "pred_predict": "1"}
  ]
}
```

9.5.7 Delete Model

```
DELETE /api/models/{model_id}
```

Permanently delete a model.

Example

```
curl -X DELETE "$BASE_URL/api/models/a7b8c9d0-e1f2-3456-7890-abcdef123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "ok": true
}
```

See also

- *Experiments API* – Viewing all models from an experiment.
- *Deployments API* – Deploying a model for serving.
- *ML Studio (What-If) API* – Running what-if scenarios against a deployed model.

9.6 Deployments API

Deployments serve trained models as real-time prediction endpoints. The MLOps lifecycle supports staging and production stages, promotion, rollback, inference logging, and data drift detection.

All deployment endpoints are prefixed with `/api/mlops`.

Endpoints

- *Create Deployment*
- *List Deployments*
- *Get Deployment Detail*
- *Predict via Deployment*
- *Promote Deployment*
- *Rollback Deployment*

- *Deactivate Deployment*
- *Drift Metrics*
- *Run Drift Detection*
- *Inference Logs*
- *Deployment Lifecycle Example*

9.6.1 Create Deployment

```
POST /api/mlops/projects/{project_id}/deployments
```

Deploy a model to either staging or production. If deploying to `production`, any existing production deployment in the same project is automatically archived.

Path Parameters

Parameter	Type	Description
`project_id`	UUID	Project that owns the model.

Request Body

Field	Type	Required	Description
`model_id`	string	Yes	UUID of the model to deploy. Must belong to the same project.
`name`	string	Yes	Deployment name.
`stage`	string	Yes	`staging` or `production`.
`traffic_percent`	integer	No	Traffic allocation percentage (1–100, default 100).
`privacy_policy_id`	string	No	Attach a privacy policy for runtime anonymization.
`privacy_anonymize_log`	boolean	No	Anonymize PII in inference logs.
`privacy_anonymize_res`	boolean	No	Anonymize PII in prediction responses.
`privacy_column_map`	object	No	Map input column names to PII types.
`privacy_threshold`	float	No	Confidence threshold for PII detection (0.0–1.0).

Example

```
curl -X POST "$BASE_URL/api/mlops/projects/d4e5f6a7-b8c9-0123-def4-567890123456/
↪deployments" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
    "name": "fraud-detector-staging",
    "stage": "staging",
    "traffic_percent": 100
  }'
```

```python
import requests

resp = requests.post(
    f"{BASE_URL}/api/mlops/projects/d4e5f6a7-b8c9-0123-def4-567890123456/deployments",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    json={
        "model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
        "name": "fraud-detector-staging",
        "stage": "staging",
        "traffic_percent": 100,
    },
)
deployment_id = resp.json()["deployment_id"]
print("Deployed:", deployment_id)
```

Response `201 Created`

```
{
  "id": "d0e1f2a3-b4c5-6789-0123-def456789012",
  "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
  "stage": "staging",
  "is_active": true
}
```

9.6.2 List Deployments

```
GET /api/mlops/projects/{project_id}/deployments
```

Return all deployments for a project.

Query Parameters

Parameter	Type	Default	Description
`limit`	integer	50	Max items (1–200).
`offset`	integer	0	Pagination offset.

Example

```
curl "$BASE_URL/api/mlops/projects/d4e5f6a7-b8c9-0123-def4-567890123456/deployments" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```python
resp = requests.get(
    f"{BASE_URL}/api/mlops/projects/d4e5f6a7-b8c9-0123-def4-567890123456/deployments",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
for dep in resp.json()["items"]:
    print(f"{dep['id']}  stage={dep['stage']}  active={dep['is_active']}")
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "d0e1f2a3-b4c5-6789-0123-def456789012",
      "model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
      "stage": "staging",
      "is_active": true,
      "config": {"name": "fraud-detector-staging", "traffic_percent": 100},
      "created_at": "2026-02-25T09:00:00Z"
    }
  ],
  "total": 1,
  "limit": 50,
  "offset": 0
}
```

9.6.3 Get Deployment Detail

```
GET /api/mlops/deployments/{deployment_id}
```

Return deployment metadata and the associated model details.

Example

```
curl "$BASE_URL/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "deployment": {
    "id": "d0e1f2a3-b4c5-6789-0123-def456789012",
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
    "stage": "staging",
    "is_active": true,
    "config": {"name": "fraud-detector-staging", "traffic_percent": 100},
    "created_at": "2026-02-25T09:00:00Z"
  },
  "model": {
    "id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
    "name": "GBM_1_AutoML",
    "algo": "GBM"
  }
}
```

9.6.4 Predict via Deployment

```
POST /api/mlops/deployments/{deployment_id}/predict
```

Send input features and receive predictions from the deployed model. This is the primary real-time prediction endpoint. Inference requests are logged automatically for monitoring and drift detection.

Request Body

Field	Type	Required	Description
`inputs`	dict or list[dict]	Yes	Feature values. Single record or batch.
`options`	object	No	Options (see table below).

Predict Options

Key	Type	Description
`include_contributions`	boolean	Include SHAP-style feature contributions (not available for StackedEnsemble models). Default: `false`.
`store_payload`	boolean	Store input payload in inference logs. Default: `true`.
`privacy_policy_id`	string	Override the deployment-level privacy policy for this request.
`anonymize_logs`	boolean	Override log anonymization for this request.
`anonymize_response`	boolean	Override response anonymization for this request.

Example

```
curl -X POST "$BASE_URL/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/
↪predict" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "inputs": {"amount": 29.99, "merchant_id": 42, "hour": 14},
    "options": {"include_contributions": true}
  }'
```

```
resp = requests.post(
    f"{BASE_URL}/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/predict",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    json={
        "inputs": {"amount": 29.99, "merchant_id": 42, "hour": 14},
        "options": {"include_contributions": True},
    },
)
result = resp.json()
print("Prediction:", result["predictions"][0]["prediction"])
```

Response `200 OK`

```
{
  "predictions": [
```

(continues on next page)

(continued from previous page)

```
    {"prediction": "0"}
  ],
  "contributions": [
    [
      {"feature": "amount", "value": -0.32},
      {"feature": "merchant_id", "value": -0.15},
      {"feature": "BiasTerm", "value": -1.20}
    ]
  ],
  "latency_ms": 45
}
```

9.6.5 Promote Deployment

```
POST /api/mlops/deployments/{deployment_id}/promote
```

Promote a deployment to production. Any existing production deployment in the same project is automatically archived.

Request Body

Field	Type	Required	Description
`to_stage`	string	Yes	Must be `production`.

Example

```
curl -X POST "$BASE_URL/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/
↪promote" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"to_stage": "production"}'
```

```
requests.post(
    f"{BASE_URL}/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/promote",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    json={"to_stage": "production"},
)
```

Response `200 OK`

```
{
  "ok": true,
  "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
  "stage": "production"
}
```

9.6.6 Rollback Deployment

```
POST /api/mlops/deployments/{deployment_id}/rollback
```

Rollback production to a previous deployment or model. Provide either `to_deployment_id` (reactivate an archived deployment) or `to_model_id` (create a new production deployment for the given model).

Request Body

Field	Type	Required	Description
`to_deployment_id`	string	No*	UUID of the archived deployment to reactivate.
`to_model_id`	string	No*	UUID of the model to deploy as the new production version.

* At least one of `to_deployment_id` or `to_model_id` is required.

Example

```
curl -X POST "$BASE_URL/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/
↪rollback" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"to_model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456"}'
```

Response `200 OK`

```
{
  "ok": true
}
```

9.6.7 Deactivate Deployment

```
POST /api/mlops/deployments/{deployment_id}/deactivate
```

Deactivate a deployment without promoting a replacement. The deployment moves to `archived` stage.

Example

```
curl -X POST "$BASE_URL/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/
↪deactivate" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "ok": true
}
```

9.6.8 Drift Metrics

```
GET /api/mlops/deployments/{deployment_id}/drift
```

Return drift detection results comparing the training distribution to recent inference data. Returns per-feature PSI scores and overall drift status.

Example

```
curl "$BASE_URL/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/drift" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/drift",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
drift = resp.json()
for feature, info in drift.get("features", {}).items():
    print(f"{feature}: PSI={info['psi']:.4f} ({info['status']})")
```

Response `200 OK`

```
{
  "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
  "overall_status": "low",
  "features": {
    "amount": {"psi": 0.023, "status": "low"},
    "merchant_id": {"psi": 0.15, "status": "moderate"},
    "hour": {"psi": 0.008, "status": "low"}
  },
  "computed_at": "2026-02-28T12:00:00Z",
  "inference_rows": 5000
}
```

9.6.9 Run Drift Detection

```
POST /api/mlops/deployments/{deployment_id}/drift/run
```

Enqueue a background job to compute drift metrics against recent inference data. The result can be retrieved via the drift metrics endpoint once the job completes.

Example

```
curl -X POST "$BASE_URL/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/drift/
↪run" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "ok": true,
  "job_id": "e1f2a3b4-c5d6-7890-1234-ef5678901234"
}
```

9.6.10 Inference Logs

```
GET /api/mlops/deployments/{deployment_id}/inference-logs
```

Return a paginated list of inference requests and responses for the deployment. Useful for monitoring, auditing, and debugging.

Query Parameters

Parameter	Type	Default	Description
`limit`	integer	50	Max items (1–500).
`offset`	integer	0	Pagination offset.
`since`	string	–	ISO 8601 timestamp. Only return logs after this time.
`until`	string	–	ISO 8601 timestamp. Only return logs before this time.

Example

```
curl "$BASE_URL/api/mlops/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/inference-
↪logs?limit=5" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "f2a3b4c5-d6e7-8901-2345-f67890123456",
      "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
      "inputs": {"amount": 29.99, "merchant_id": 42, "hour": 14},
      "prediction": "0",
      "latency_ms": 45,
      "status_code": 200,
      "created_at": "2026-02-28T14:30:00Z"
    }
  ],
  "total": 5000,
  "limit": 5,
  "offset": 0
}
```

Deployment Stages

Stage	Description
`staging`	Pre-production testing. Use this to validate model behavior before going live.
`production`	Live serving. Predictions from this deployment are production traffic.

Deployment Status Values

Status	Description
`active`	Deployment is running and serving predictions.
`inactive`	Deployment has been deactivated. Not serving predictions.
`failed`	Deployment failed to initialize or load the model.

Alert Metrics Reference

Metric	Description
`drift_psi`	Population Stability Index. Measures feature distribution drift between training and production data.
`accuracy_degradation`	Degradation in prediction accuracy over time.
`error_rate`	Percentage of predictions returning errors.
`latency_p99`	99th percentile prediction latency in milliseconds.
`model_staleness`	Days since the model was last retrained.
`prediction_anomaly`	Anomalous prediction patterns detected.

Alert Condition Operators

Operator	Symbol	Description
`gt`	`>`	Greater than threshold.
`lt`	`<`	Less than threshold.
`gte`	`>=`	Greater than or equal to threshold.
`lte`	`<=`	Less than or equal to threshold.
`eq`	`==`	Equal to threshold.
`neq`	`!=`	Not equal to threshold.

Alert Severity Levels

Severity	Description
`info`	Informational alert. No action required.
`warning`	Warning. Should be investigated.
`critical`	Critical issue requiring immediate attention.

Notification Channels

Channel	Description
`email`	Send alert to one or more email addresses.
`slack`	Post alert to a Slack channel via webhook.
`webhook`	Send alert payload to a custom HTTP endpoint.

Alert Rule Options

Option	Type	Description
`cooldown_minutes`	integer	Minimum minutes between repeated alerts (1–1440, default 60).
`escalation_minutes`	integer	Minutes before escalating severity (optional, 1–1440).
`escalation_severity`	string	Severity to escalate to: `warning` or `critical`.

9.6.11 Deployment Lifecycle Example

The following shows the full lifecycle: create a staging deployment, test it, promote to production, and rollback if needed.

```
import requests

BASE_URL = "http://localhost:8888"
HEADERS = {"Authorization": "Bearer YOUR_API_KEY"}
PROJECT_ID = "d4e5f6a7-b8c9-0123-def4-567890123456"
MODEL_ID = "a7b8c9d0-e1f2-3456-7890-abcdef123456"

# 1. Deploy to staging
resp = requests.post(
    f"{BASE_URL}/api/mlops/projects/{PROJECT_ID}/deployments",
    headers=HEADERS,
    json={"model_id": MODEL_ID, "name": "v2-staging", "stage": "staging"},
)
deployment_id = resp.json()["deployment_id"]

# 2. Test with a sample prediction
resp = requests.post(
    f"{BASE_URL}/api/mlops/deployments/{deployment_id}/predict",
    headers=HEADERS,
    json={"inputs": {"amount": 50.0, "merchant_id": 10, "hour": 12}},
)
assert resp.status_code == 200
print("Staging test passed:", resp.json()["predictions"])

# 3. Promote to production
requests.post(
    f"{BASE_URL}/api/mlops/deployments/{deployment_id}/promote",
    headers=HEADERS,
    json={"to_stage": "production"},
)
print("Promoted to production")

# 4. Later: rollback if needed
# requests.post(
#     f"{BASE_URL}/api/mlops/deployments/{deployment_id}/rollback",
#     headers=HEADERS,
#     json={"to_model_id": OLD_MODEL_ID},
# )
```

See also

- *Models API* – Model details and direct prediction.
- *ML Studio (What-If) API* – What-if analysis against a deployment.
- *Reports API* – Generate deployment reports.

9.7 Reports API

Reports are generated asynchronously as background jobs. They produce HTML or PDF documents summarizing experiments, models, projects, deployments, or SynthGen runs. Reports can optionally include AI-generated insights.

Endpoints

- *Create Report*
- *List Reports*
- *Get Report Detail*
- *View Report Inline*
- *Download Report*

9.7.1 Create Report

```
POST /api/reports
```

Create a report and enqueue a `report_generate` background job. The report starts in `queued` status and transitions to `succeeded` once the artifact is ready.

Request Body

Field	Type	Required	Description
`project_id`	string	Yes	UUID of the project.
`kind`	string	Yes	Report type: `experiment`, `model`, `project`, `deployment`, or `synthgen`.
`entity_id`	string	Yes	UUID of the entity to report on (the experiment, model, deployment, etc.).
`options`	object	No	Report customization (see below).

Options

Key	Type	Description
`llm_insights`	boolean	Include AI-generated commentary and recommendations. Default: `false`.
`title`	string	Custom title for the report.
`comments`	string	Free-form text added as a comments section.
`include_metrics`	boolean	Include performance metrics table. Default: `true`.
`include_charts`	boolean	Include visualizations (ROC curve, variable importance). Default: `true`.
`include_hyperparamete`	boolean	Include model hyperparameters. Default: `true`.
`include_data_summary`	boolean	Include dataset statistics. Default: `true`.

Example

```
curl -X POST "$BASE_URL/api/reports" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "kind": "experiment",
    "entity_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
    "options": {
      "llm_insights": true,
      "title": "Fraud Detection Model Report",
      "comments": "Q4 quarterly review"
    }
  }'
```

```
import requests
import time

# Create report
resp = requests.post(f"{BASE_URL}/api/reports", headers={
    "Authorization": "Bearer YOUR_API_KEY",
}, json={
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "kind": "experiment",
    "entity_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
    "options": {"llm_insights": True},
})
report_id = resp.json()["report_id"]

# Poll until ready
while True:
    r = requests.get(f"{BASE_URL}/api/reports/{report_id}", headers={
        "Authorization": "Bearer YOUR_API_KEY",
    })
    status = r.json()["report"]["status"]
    if status in ("succeeded", "failed"):
        break
    time.sleep(5)

# Download the report
```

(continues on next page)

(continued from previous page)

```
if status == "succeeded":
    dl = requests.get(f"{BASE_URL}/api/reports/{report_id}/download", headers={
        "Authorization": "Bearer YOUR_API_KEY",
    })
    with open("report.pdf", "wb") as f:
        f.write(dl.content)
```

Response `201 Created`

```
{
  "report_id": "e1f2a3b4-c5d6-7890-1234-ef5678901234",
  "job_id": "f2a3b4c5-d6e7-8901-2345-f67890123456",
  "status": "queued"
}
```

Report Kinds

Kind	Entity	Description
`experiment`	Experiment ID	AutoML training summary: leaderboard, metrics, feature importance, model comparison charts.
`model`	Model ID	Individual model analysis: hyperparameters, performance metrics, feature importance, learning curves.
`project`	Project ID	Project overview: all experiments, datasets, models, and deployment status.
`synthgen`	SynthGen Job ID	Synthetic data quality: distribution comparison, correlation preservation, privacy metrics.
`deployment`	Deployment ID	Deployment health: prediction volume, latency, drift metrics, error rates.

Note

Privacy PDF reports are generated with `GET /api/privacy/sessions/{session_id}/report` (Privacy API), not via `POST /api/reports`.

Report Options

Options vary by report kind. Common options:

Option	Type	Description
`llm_insights`	boolean	Enable AI-powered recommendations using LLM analysis. Requires `OPENAI_API_KEY` configured on the server. Default `false`.
`include_leaderboard`	boolean	Include model leaderboard table (experiment reports).
`include_feature_importance`	boolean	Include feature importance charts (experiment/model reports).
`include_confusion_matrix`	boolean	Include confusion matrix (classification reports).
`include_roc_curve`	boolean	Include ROC curve chart (classification reports).
`include_learning_curves`	boolean	Include learning curve charts (model reports).

9.7.2 List Reports

```
GET /api/reports
```

Query Parameters

Parameter	Type	Default	Description
`project_id`	string	–	**Required.** Filter by project.
`kind`	string	–	Filter by report type.
`entity_id`	string	–	Filter by entity.
`limit`	integer	50	Max items (1–500).
`offset`	integer	0	Pagination offset.

Example

```
curl "$BASE_URL/api/reports?project_id=d4e5f6a7-b8c9-0123-def4-567890123456&
↪kind=experiment" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "e1f2a3b4-c5d6-7890-1234-ef5678901234",
      "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
      "kind": "experiment",
      "entity_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
      "status": "succeeded",
      "summary": {},
      "artifact_id": "a3b4c5d6-e7f8-9012-3456-789012345678",
      "created_at": "2026-02-28T15:00:00Z"
    }
  ],
  "total": 1,
  "limit": 50,
  "offset": 0
}
```

9.7.3 Get Report Detail

```
GET /api/reports/{report_id}
```

Return the report metadata and associated job ID.

Example

```
curl "$BASE_URL/api/reports/e1f2a3b4-c5d6-7890-1234-ef5678901234" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "report": {
    "id": "e1f2a3b4-c5d6-7890-1234-ef5678901234",
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "kind": "experiment",
    "entity_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
    "status": "succeeded",
    "summary": {},
    "artifact_id": "a3b4c5d6-e7f8-9012-3456-789012345678",
    "job_id": "f2a3b4c5-d6e7-8901-2345-f67890123456"
  }
}
```

9.7.4 View Report Inline

```
GET /api/reports/{report_id}/view
```

View the report inline in the browser. Returns the rendered HTML or PDF with an `inline` Content-Disposition header. Suitable for embedding in an iframe.

Returns `409 Conflict` if the report is still processing.

Example

```
# Open in browser
curl "$BASE_URL/api/reports/e1f2a3b4-c5d6-7890-1234-ef5678901234/view" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -o report_preview.html
```

Response `200 OK`

Returns `text/html` or `application/pdf` depending on the report format.

9.7.5 Download Report

```
GET /api/reports/{report_id}/download
```

Download the report as an attachment (PDF or HTML file).

Returns `409 Conflict` if the report is still processing.

Example

```
curl -o report.pdf \
  "$BASE_URL/api/reports/e1f2a3b4-c5d6-7890-1234-ef5678901234/download" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/reports/e1f2a3b4-c5d6-7890-1234-ef5678901234/download",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
```

(continues on next page)

(continued from previous page)

```
with open("report.pdf", "wb") as f:
    f.write(resp.content)
```

Response `200 OK`

Binary file download with `Content-Disposition: attachment`.

See also

- *Experiments API* – Experiment reports with model comparisons.
- *Deployments API* – Deployment performance reports.
- *Privacy Suite API* – Privacy session reports.

9.8 Privacy Suite API

The Privacy Suite detects and transforms personally identifiable information (PII) in datasets. It supports 72+ PII types, four compliance profiles (HIPAA, GDPR, PCI-DSS, CCPA), and seven transformation actions.

The typical workflow is: **create policy** (with rules) then **create session** (linking policy to dataset) then **detect** then **transform** then **download**.

All endpoints are prefixed with `/api/privacy`.

Endpoints

- *Create Policy*
- *List Policies*
- *Get Policy*
- *Update Policy*
- *Delete Policy*
- *Create Detection Rule*
- *List Rules*
- *Create Session*
- *List Sessions*
- *Get Session Detail*
- *Run PII Detection*
- *Apply Anonymization*
- *Detection Results*
- *Download Anonymized Data*
- *Generate Privacy Report*

- *List PII Types*
- *List Actions*
- *List Compliance Profiles*
- *Full Workflow Example*
- *Supported PII Types*
- *Transformation Actions Reference*
- *Compliance Profiles*
- *Detection Confidence Levels*

9.8.1 Create Policy

```
POST /api/privacy/policies
```

Create a privacy policy that defines which PII types to detect and how to handle them.

Request Body

Field	Type	Required	Description
`project_id`	string	Yes	UUID of the project.
`name`	string	Yes	Policy name.
`description`	string	No	Optional description.
`profile`	string	No	Compliance profile: `hipaa`, `gdpr`, `pci_dss`, `ccpa`, or `custom`.
`status`	string	No	`draft` (default) or `active`.
`settings`	object	No	Additional configuration (e.g. sensitivity thresholds).

Example

```bash
curl -X POST "$BASE_URL/api/privacy/policies" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "name": "HIPAA Compliance Policy",
    "description": "Detect and redact all 18 PHI identifiers",
    "profile": "hipaa",
    "status": "active"
  }'
```

```python
import requests

resp = requests.post(f"{BASE_URL}/api/privacy/policies", headers={
    "Authorization": "Bearer YOUR_API_KEY",
}, json={
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
```

(continues on next page)

(continued from previous page)

```
    "name": "HIPAA Compliance Policy",
    "description": "Detect and redact all 18 PHI identifiers",
    "profile": "hipaa",
    "status": "active",
})
policy_id = resp.json()["policy_id"]
```

Response `201 Created`

```
{
  "id": "1a2b3c4d-5e6f-7890-abcd-ef1234567890",
  "policy_id": "1a2b3c4d-5e6f-7890-abcd-ef1234567890"
}
```

9.8.2 List Policies

```
GET /api/privacy/policies
```

Query Parameters

Parameter	Type	Default	Description
`project_id`	string	–	Filter by project. If omitted, returns all accessible policies.
`status`	string	–	Filter by status (`draft`, `active`).
`limit`	integer	50	Max items.
`offset`	integer	0	Pagination offset.

Example

```
curl "$BASE_URL/api/privacy/policies?project_id=d4e5f6a7-b8c9-0123-def4-567890123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "1a2b3c4d-5e6f-7890-abcd-ef1234567890",
      "name": "HIPAA Compliance Policy",
      "profile": "hipaa",
      "status": "active",
      "rules_count": 5,
      "compliance_score": 100,
      "detected_count": 4,
      "protected_count": 4,
      "pending_count": 0,
      "created_at": "2026-02-20T09:00:00Z"
    }
  ],
  "total": 1,
```

(continues on next page)

(continued from previous page)

```
  "limit": 50,
  "offset": 0
}
```

9.8.3 Get Policy

```
GET /api/privacy/policies/{policy_id}
```

Return policy details including its rules and compliance metrics.

Example

```
curl "$BASE_URL/api/privacy/policies/1a2b3c4d-5e6f-7890-abcd-ef1234567890" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "policy": {
    "id": "1a2b3c4d-5e6f-7890-abcd-ef1234567890",
    "name": "HIPAA Compliance Policy",
    "profile": "hipaa",
    "project_name": "Fraud Detection v2",
    "rules_count": 5,
    "compliance_score": 100
  },
  "rules": [
    {
      "id": "2b3c4d5e-6f7a-8901-bcde-f23456789012",
      "pii_type": "email",
      "action": "redact",
      "is_enabled": true,
      "priority": 10,
      "config": {}
    }
  ]
}
```

9.8.4 Update Policy

```
PATCH /api/privacy/policies/{policy_id}
```

Partial update of a policy. Only provided fields are modified.

Request Body

Field	Type	Required	Description
`name`	string	No	Updated name.
`description`	string	No	Updated description.
`status`	string	No	`draft` or `active`.
`profile`	string	No	Updated compliance profile.
`settings`	object	No	Updated configuration.

Example

```
curl -X PATCH "$BASE_URL/api/privacy/policies/1a2b3c4d-5e6f-7890-abcd-ef1234567890" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{"status": "active"}'
```

Response `200 OK`

```
{
  "ok": true
}
```

9.8.5 Delete Policy

```
DELETE /api/privacy/policies/{policy_id}
```

Permanently delete a privacy policy and its rules.

Example

```
curl -X DELETE "$BASE_URL/api/privacy/policies/1a2b3c4d-5e6f-7890-abcd-ef1234567890" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "ok": true
}
```

9.8.6 Create Detection Rule

```
POST /api/privacy/policies/{policy_id}/rules
```

Add a PII detection and transformation rule to a policy.

Request Body

Field	Type	Required	Description
`pii_type`	string	Yes	PII type identifier (e.g. `email`, `ssn`, `credit_card`). Use `GET /api/privacy/pii-types` for the full list.
`action`	string	Yes	Transformation action: `mask`, `hash`, `redact`, `encrypt`, `generalize`, `suppress`, or `pseudonymize`.
`is_enabled`	boolean	No	Whether this rule is active (default `true`).
`priority`	integer	No	Higher priority rules take precedence (default `0`).
`config`	object	No	Action-specific configuration (e.g. mask character, hash algorithm).

Example

```
curl -X POST "$BASE_URL/api/privacy/policies/1a2b3c4d-5e6f-7890-abcd-ef1234567890/rules"␣
↪\
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "pii_type": "email",
    "action": "redact",
    "is_enabled": true,
    "priority": 10
  }'
```

```
resp = requests.post(
    f"{BASE_URL}/api/privacy/policies/1a2b3c4d-5e6f-7890-abcd-ef1234567890/rules",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    json={
        "pii_type": "email",
        "action": "redact",
        "is_enabled": True,
        "priority": 10,
    },
)
rule_id = resp.json()["rule_id"]
```

Response `201 Created`

```
{
  "id": "2b3c4d5e-6f7a-8901-bcde-f23456789012",
  "rule_id": "2b3c4d5e-6f7a-8901-bcde-f23456789012"
}
```

9.8.7 List Rules

```
GET /api/privacy/policies/{policy_id}/rules
```

Return all rules for a policy.

Example

```
curl "$BASE_URL/api/privacy/policies/1a2b3c4d-5e6f-7890-abcd-ef1234567890/rules" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`


```
{
  "rules": [
    {
      "id": "2b3c4d5e-6f7a-8901-bcde-f23456789012",
      "policy_id": "1a2b3c4d-5e6f-7890-abcd-ef1234567890",
      "pii_type": "email",
      "action": "redact",
      "is_enabled": true,
      "enabled": true,
      "priority": 10,
      "config": {}
    },
    {
      "id": "3c4d5e6f-7a8b-9012-cdef-345678901234",
      "policy_id": "1a2b3c4d-5e6f-7890-abcd-ef1234567890",
      "pii_type": "ssn",
      "action": "hash",
      "is_enabled": true,
      "enabled": true,
      "priority": 20,
      "config": {"algorithm": "sha256"}
    }
  ]
}
```


9.8.8 Create Session

```
POST /api/privacy/sessions
```

Create a privacy session that links a policy to a dataset version for PII detection and transformation.

Request Body

Field	Type	Required	Description
`policy_id`	string	Yes	UUID of the privacy policy to apply.
`dataset_id`	string	No*	UUID of the dataset (uses latest version).
`dataset_version_id`	string	No*	UUID of a specific dataset version.
`project_id`	string	No	Must match the policy's project if provided.
`sample_size`	integer	No	Number of rows to sample for detection (null = all rows).
`deep_scan`	boolean	No	Enable deep scan mode for more thorough detection (default `false`).

* Provide either `dataset_id` or `dataset_version_id`.

Example

```
curl -X POST "$BASE_URL/api/privacy/sessions" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "policy_id": "1a2b3c4d-5e6f-7890-abcd-ef1234567890",
    "dataset_id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
    "deep_scan": true
  }'
```

Response `201 Created`

```
{
  "id": "4d5e6f7a-8b9c-0123-def4-567890123456",
  "session_id": "4d5e6f7a-8b9c-0123-def4-567890123456"
}
```

9.8.9 List Sessions

```
GET /api/privacy/sessions
```

Query Parameters

Parameter	Type	Default	Description
`project_id`	string	–	Filter by project.
`policy_id`	string	–	Filter by policy.
`limit`	integer	50	Max items.
`offset`	integer	0	Pagination offset.

Example

```
curl "$BASE_URL/api/privacy/sessions?project_id=d4e5f6a7-b8c9-0123-def4-567890123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "4d5e6f7a-8b9c-0123-def4-567890123456",
      "policy_name": "HIPAA Compliance Policy",
      "dataset_name": "Transactions Q4",
      "status": "completed",
      "pii_found": 4,
      "transformed": 4,
      "created_at": "2026-02-25T10:00:00Z"
    }
  ],
  "total": 1,
  "limit": 50,
  "offset": 0
}
```

9.8.10 Get Session Detail

```
GET /api/privacy/sessions/{session_id}
```

Return session details including scan progress, detection counts, and transformation status.

Example

```
curl "$BASE_URL/api/privacy/sessions/4d5e6f7a-8b9c-0123-def4-567890123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "session": {
    "id": "4d5e6f7a-8b9c-0123-def4-567890123456",
    "policy_name": "HIPAA Compliance Policy",
    "dataset_name": "Transactions Q4",
    "status": "completed",
    "rows_scanned": 50000,
    "columns_analyzed": 15,
    "pii_columns": 4,
    "pii_instances": 200000,
    "transformed": 4,
    "pending": 0,
    "progress": 100,
    "progress_message": "Transformation complete",
    "output_artifact_id": "b4c5d6e7-f8a9-0123-4567-890abcdef123"
  }
}
```

9.8.11 Run PII Detection

```
POST /api/privacy/sessions/{session_id}/detect
```

Enqueue a `privacy_detect` background job to scan the dataset for PII. The session status transitions to `detecting`.

Example

```
curl -X POST "$BASE_URL/api/privacy/sessions/4d5e6f7a-8b9c-0123-def4-567890123456/detect
↪" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "job_id": "c5d6e7f8-a9b0-1234-5678-90abcdef1234",
  "session_id": "4d5e6f7a-8b9c-0123-def4-567890123456",
  "status": "detecting"
}
```

9.8.12 Apply Anonymization

```
POST /api/privacy/sessions/{session_id}/transform
```

Enqueue a `privacy_transform` background job to apply the transformation rules to detected PII. The session status transitions to `transforming` and then `completed`.

Example

```
curl -X POST "$BASE_URL/api/privacy/sessions/4d5e6f7a-8b9c-0123-def4-567890123456/
↪transform" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "job_id": "d6e7f8a9-b0c1-2345-6789-0abcdef12345",
  "session_id": "4d5e6f7a-8b9c-0123-def4-567890123456",
  "status": "transforming"
}
```

9.8.13 Detection Results

```
GET /api/privacy/sessions/{session_id}/results
```

Return aggregated PII detection results grouped by column and PII type.

Query Parameters

Parameter	Type	Default	Description
`pii_type`	string	–	Filter by PII type (e.g. `email`).
`confidence`	string	–	Filter by confidence bucket: `high` (>=0.9), `medium` (0.7–0.9), `low` (<0.7).

Example

```
curl "$BASE_URL/api/privacy/sessions/4d5e6f7a-8b9c-0123-def4-567890123456/results" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/privacy/sessions/4d5e6f7a-8b9c-0123-def4-567890123456/results",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
for r in resp.json()["results"]:
    print(f"  {r['column']}: {r['pii_type']} (confidence={r['confidence']:.2f}, "
          f"instances={r['instances']}, action={r['action']})")
```

Response `200 OK`

```
{
  "results": [
```

(continues on next page)

(continued from previous page)

```
    {
      "column": "customer_email",
      "pii_type": "email",
      "confidence": 0.98,
      "instances": 49500,
      "sample": "alice@example.com",
      "action": "redact",
      "transformed": true,
      "samples": []
    },
    {
      "column": "customer_name",
      "pii_type": "name",
      "confidence": 0.92,
      "instances": 48000,
      "sample": "Alice Chen",
      "action": "mask",
      "transformed": true,
      "samples": []
    }
  ]
}
```

9.8.14 Download Anonymized Data

```
GET /api/privacy/sessions/{session_id}/download
```

Download the anonymized dataset. Redirects (302) to the artifact download URL.

Returns `404` if the transformation has not been completed yet.

Example

```
curl -L -o anonymized_data.csv \
  "$BASE_URL/api/privacy/sessions/4d5e6f7a-8b9c-0123-def4-567890123456/download" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/privacy/sessions/4d5e6f7a-8b9c-0123-def4-567890123456/download",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    allow_redirects=True,
)
with open("anonymized_data.csv", "wb") as f:
    f.write(resp.content)
```

Response `302 Found`

Redirects to `/api/artifacts/{artifact_id}/download`.

9.8.15 Generate Privacy Report

```
GET /api/privacy/sessions/{session_id}/report
```

Generate and download a branded PDF report summarizing the privacy session: policy details, detected PII, applied transformations, and compliance status.

Query Parameters

Parameter	Type	Default	Description
`ai_insights`	boolean	`false`	Include AI-generated commentary.

Example

```
curl "$BASE_URL/api/privacy/sessions/4d5e6f7a-8b9c-0123-def4-567890123456/report?ai_
↪insights=true" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "artifact_id": "e7f8a9b0-c1d2-3456-7890-abcdef123456",
  "download_url": "/api/artifacts/e7f8a9b0-c1d2-3456-7890-abcdef123456/download"
}
```

9.8.16 List PII Types

```
GET /api/privacy/pii-types
```

Return the full list of supported PII types. Over 72 types covering names, addresses, phone numbers, financial identifiers, medical records, biometric data, and more.

Example

```
curl "$BASE_URL/api/privacy/pii-types" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "types": [
    {"id": "email", "name": "Email"},
    {"id": "phone", "name": "Phone"},
    {"id": "ssn", "name": "Ssn"},
    {"id": "credit_card", "name": "Credit Card"},
    {"id": "name", "name": "Name"},
    {"id": "address", "name": "Address"},
    {"id": "date_of_birth", "name": "Date Of Birth"},
    {"id": "ip_address", "name": "Ip Address"},
    {"id": "medical_record_number", "name": "Medical Record Number"}
  ]
}
```

9.8.17 List Actions

```
GET /api/privacy/actions
```

Return all supported transformation actions.

Example

```
curl "$BASE_URL/api/privacy/actions" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "actions": [
    {"id": "mask", "name": "Mask"},
    {"id": "hash", "name": "Hash"},
    {"id": "redact", "name": "Redact"},
    {"id": "encrypt", "name": "Encrypt"},
    {"id": "generalize", "name": "Generalize"},
    {"id": "suppress", "name": "Suppress"},
    {"id": "pseudonymize", "name": "Pseudonymize"}
  ]
}
```

9.8.18 List Compliance Profiles

```
GET /api/privacy/profiles
```

Return available compliance profiles with their default PII types and recommended actions.

Example

```
curl "$BASE_URL/api/privacy/profiles" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "profiles": [
    {
      "id": "hipaa",
      "name": "HIPAA",
      "description": "Health Insurance Portability and Accountability Act - 18 PHI␣
↪identifiers",
      "pii_types": ["name", "address", "date", "phone", "fax", "email", "ssn",
                    "medical_record_number", "health_plan_id", "account_number",
                    "certificate_license", "vehicle_id", "device_id", "url",
                    "ip_address", "biometric", "photo", "other_unique_id"],
      "default_action": "redact"
```

(continues on next page)

(continued from previous page)

```
    },
    {
      "id": "gdpr",
      "name": "GDPR",
      "description": "General Data Protection Regulation - EU personal data protection",
      "pii_types": ["name", "email", "phone", "address", "date_of_birth",
                    "national_id", "passport", "ip_address", "cookie_id",
                    "location", "biometric", "genetic", "health", "political",
                    "religious", "sexual_orientation", "trade_union"],
      "default_action": "pseudonymize"
    },
    {
      "id": "pci_dss",
      "name": "PCI-DSS",
      "description": "Payment Card Industry Data Security Standard",
      "pii_types": ["credit_card", "cvv", "pin", "magnetic_stripe",
                    "cardholder_name", "expiration_date", "service_code"],
      "default_action": "tokenize"
    },
    {
      "id": "ccpa",
      "name": "CCPA",
      "description": "California Consumer Privacy Act",
      "pii_types": ["name", "address", "email", "phone", "ssn", "drivers_license",
                    "passport", "ip_address", "browsing_history", "purchase_history",
                    "biometric", "geolocation", "employment_info", "education_info"],
      "default_action": "redact"
    },
    {
      "id": "custom",
      "name": "Custom",
      "description": "Create your own custom privacy profile",
      "pii_types": [],
      "default_action": "mask"
    }
  ]
}
```

9.8.19 Full Workflow Example

The following demonstrates the complete PII detection and anonymization workflow:

```
import requests
import time

BASE_URL = "http://localhost:8888"
HEADERS = {"Authorization": "Bearer YOUR_API_KEY"}
PROJECT_ID = "d4e5f6a7-b8c9-0123-def4-567890123456"
DATASET_ID = "e5f6a7b8-c9d0-1234-5678-90abcdef1234"
```

(continues on next page)

(continued from previous page)

```python
# 1. Create a HIPAA policy
resp = requests.post(f"{BASE_URL}/api/privacy/policies", headers=HEADERS, json={
    "project_id": PROJECT_ID,
    "name": "HIPAA Policy",
    "profile": "hipaa",
    "status": "active",
})
policy_id = resp.json()["policy_id"]

# 2. Add detection rules
for pii_type in ["email", "ssn", "phone", "name"]:
    requests.post(
        f"{BASE_URL}/api/privacy/policies/{policy_id}/rules",
        headers=HEADERS,
        json={"pii_type": pii_type, "action": "redact", "is_enabled": True},
    )

# 3. Create a session linking the policy to the dataset
resp = requests.post(f"{BASE_URL}/api/privacy/sessions", headers=HEADERS, json={
    "policy_id": policy_id,
    "dataset_id": DATASET_ID,
    "deep_scan": True,
})
session_id = resp.json()["session_id"]

# 4. Run PII detection
requests.post(
    f"{BASE_URL}/api/privacy/sessions/{session_id}/detect",
    headers=HEADERS,
)

# Poll until detection completes
while True:
    r = requests.get(
        f"{BASE_URL}/api/privacy/sessions/{session_id}",
        headers=HEADERS,
    )
    status = r.json()["session"]["status"]
    if status not in ("pending", "detecting"):
        break
    time.sleep(5)

# 5. View detection results
results = requests.get(
    f"{BASE_URL}/api/privacy/sessions/{session_id}/results",
    headers=HEADERS,
).json()
for r in results["results"]:
    print(f"Found {r['pii_type']} in column '{r['column']}' "
          f"({r['instances']} instances, confidence={r['confidence']:.0%})")

# 6. Apply anonymization
```

(continues on next page)

(continued from previous page)

```
requests.post(
    f"{BASE_URL}/api/privacy/sessions/{session_id}/transform",
    headers=HEADERS,
)

# Poll until transformation completes
while True:
    r = requests.get(
        f"{BASE_URL}/api/privacy/sessions/{session_id}",
        headers=HEADERS,
    )
    status = r.json()["session"]["status"]
    if status not in ("transforming",):
        break
    time.sleep(5)

# 7. Download the anonymized dataset
resp = requests.get(
    f"{BASE_URL}/api/privacy/sessions/{session_id}/download",
    headers=HEADERS,
    allow_redirects=True,
)
with open("anonymized_data.csv", "wb") as f:
    f.write(resp.content)
print("Anonymized dataset saved.")
```

9.8.20 Supported PII Types

The Privacy Suite supports 38 pattern-detected PII types organized into seven categories. Each type is identified by a unique string ID used in detection rules and API responses.

Personal Identity

PII Type ID	Description
`name`	Personal names (via context detection).
`ssn`	US Social Security Numbers (xxx-xx-xxxx).
`date_of_birth`	Date of birth patterns.
`passport_us`	US passport numbers.
`passport_uk`	UK passport numbers.
`passport_eu`	EU passport numbers.
`drivers_license_us`	US driver's license numbers.
`national_id_uk`	UK National Insurance numbers.
`national_id_ca`	Canadian Social Insurance numbers.
`ein`	Employer Identification Numbers.
`itin`	Individual Taxpayer Identification Numbers.
`vin`	Vehicle Identification Numbers.

Contact Information

PII Type ID	Description
email	Email addresses (RFC 5322).
phone_us	US phone numbers (xxx-xxx-xxxx, (xxx) xxx-xxxx).
phone_intl	International phone numbers (+country code).
phone_uk	UK phone numbers.
phone_eu	EU phone numbers.
address	Physical addresses (via context detection).

Financial

PII Type ID	Description
credit_card	Credit card numbers (Visa, Mastercard, Discover).
credit_card_amex	American Express card numbers.
iban	International Bank Account Numbers.
swift_bic	SWIFT/BIC bank codes.
routing_number	US bank routing numbers.

Healthcare (HIPAA)

PII Type ID	Description
medical_record	Medical record numbers (HIPAA).
dea_number	DEA registration numbers.
npi	National Provider Identifiers.
health_plan_id	Health plan beneficiary numbers.

Geographic / Location

PII Type ID	Description
zip_code_us	US ZIP codes (5-digit and ZIP+4).
postal_code_uk	UK postal codes.
postal_code_ca	Canadian postal codes.
latitude	Geographic latitude values.
longitude	Geographic longitude values.
geo_coordinate	Geographic coordinate pairs.

Digital Identifiers

PII Type ID	Description
ip_address	IPv4 addresses.
ipv6_address	IPv6 addresses.
mac_address	MAC addresses.
url	URLs (http/https).
uuid	UUID v4 identifiers.

Cryptocurrency

PII Type ID	Description
`bitcoin_address`	Bitcoin wallet addresses.
`ethereum_address`	Ethereum wallet addresses.

9.8.21 Transformation Actions Reference

All eleven transformation actions supported by the Privacy Suite are listed below. Specify the action ID when creating detection rules. Some actions accept additional configuration via the `config` object.

Action ID	Description	Config Options
`redact`	Replace detected value with `[REDACTED]`.	None.
`mask`	Partial masking (e.g., `***-**-1234`).	`show_last` (int) – number of trailing characters to reveal.
`hash`	One-way SHA-256 hash. Preserves referential integrity across columns so the same input always produces the same hash.	None.
`encrypt`	AES-256 reversible encryption. The original value can be recovered with the encryption key.	Requires an encryption key configured in policy settings.
`pseudonymize`	Replace with consistent fake values (e.g., fake names, emails). The same input always maps to the same pseudonym within a session.	None.
`tokenize`	Replace with random tokens. A lookup table is maintained so tokens can be reversed if needed.	None.
`generalize`	Reduce precision of the value.	`to` (string) – target granularity: `"year"`, `"month"`, `"age_range"`, or `"region"`.
`suppress`	Remove the column entirely from output.	None.
`truncate`	Shorten values to a fixed length.	`length` (int) – maximum character length.
`k_anonymize`	Apply k-anonymity grouping. Rows are grouped so that each combination of quasi-identifiers appears at least k times.	`k` (int) – minimum group size.
`differential_priv`	Add calibrated statistical noise to numeric values.	`epsilon` (float) – privacy budget. Lower values provide stronger privacy but more noise.

9.8.22 Compliance Profiles

The Privacy Suite ships with seven built-in compliance profiles. Each profile pre-selects the PII types mandated by the corresponding regulation and assigns a recommended default action.

Profile ID	Regulation	Scope
`hipaa`	HIPAA (Health Insurance Portability and Accountability Act)	Targets all 18 PHI identifier categories: names, geographic data, dates, phone numbers, fax numbers, email addresses, SSNs, medical record numbers, health plan IDs, account numbers, certificate / license numbers, vehicle IDs, device serial numbers, URLs, IP addresses, biometric identifiers, photographs, and other unique identifiers.
`gdpr`	GDPR (EU General Data Protection Regulation 2016/679)	All personal data as defined by the regulation: names, emails, IP addresses, location data, online identifiers, genetic data, and biometric data.
`pci_dss`	PCI DSS v4.0 (Payment Card Industry Data Security Standard)	Card numbers, CVVs, cardholder names, expiration dates, PINs, and authentication data.
`ccpa`	CCPA (California Consumer Privacy Act)	Broad PII scope: personal identifiers, commercial information, biometrics, internet activity, geolocation, and professional information.
`financial`	General Financial Compliance	Account numbers, routing numbers, IBAN, SWIFT/BIC codes, EINs, and tax identification numbers.
`sox`	SOX (Sarbanes-Oxley Act)	Financial records, audit trails, and internal control documentation.
`glba`	GLBA (Gramm-Leach-Bliley Act)	Consumer financial data including account numbers, SSNs, income records, and credit history.

In addition to these built-in profiles, you can select `custom` to create a profile with any combination of PII types and actions.

9.8.23 Detection Confidence Levels

Each PII detection is assigned a confidence score between `0.0` and `1.0` that reflects how certain the detector is about the match.

Confidence Score Ranges

Score	Detection Method
`0.95`	Exact column name match (e.g., a column named `email` or `ssn`).
`0.85`	Contains-keyword match (e.g., a column named `customer_email_addr`).
`0.80`	Prefix or suffix match (e.g., a column ending in `_phone`).
`0.60 -- 0.95`	Pattern / regex match. The exact score varies by pattern specificity.

The default confidence threshold is **0.8 (80%)**. Detections below this threshold are not reported. You can lower the threshold in the policy settings:

```
{
  "settings": {
    "confidence_threshold": 0.7
  }
}
```

Violation Severity Levels

When a compliance profile is active, each unresolved detection is classified into one of three severity levels:

Severity	Meaning
`error`	Critical compliance violation that must be resolved before the dataset can be considered compliant.
`warning`	Non-critical issue that should be reviewed. May indicate a borderline detection or a lower-priority PII type.
`info`	Informational finding. Logged for audit purposes but does not affect the compliance score.

See also

- *Datasets API* – Uploading datasets for PII scanning.
- *Deployments API* – Attaching privacy policies to deployments.
- *Reports API* – Generating compliance reports.

9.9 SynthGen API

SynthGen generates synthetic tabular data that preserves the statistical properties of your original dataset. It supports four model architectures: CTGAN, CopulaGAN, TVAE, and Gaussian Copula.

All endpoints are prefixed with `/api/synthgen`.

Endpoints

- *Create Model*
- *List Models*
- *Get Model Detail*
- *Generate Synthetic Data*
- *Delete Model*
- *List Jobs*
- *Get Job Detail*
- *Download Synthetic Data*
- *List Model Types*
- *Full Workflow Example*

9.9.1 Create Model

```
POST /api/synthgen/models
```

Create a new SynthGen model and enqueue a `synthgen_train` background job. Training begins automatically after the job is picked up by the worker.

Request Body

Field	Type	Required	Description
`project_id`	string	Yes	UUID of the project.
`dataset_version_id`	string	Yes	UUID of the dataset version to train on.
`name`	string	Yes	Model name.
`model_type`	string	No	Model architecture: `ctgan` (default), `copulagan`, `tvae`, or `gaussian_copula`.
`config`	object	No	Architecture-specific hyperparameters (e.g. `epochs`, `batch_size`, `embedding_dim`).

Example

```
curl -X POST "$BASE_URL/api/synthgen/models" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
    "name": "Transactions CTGAN",
    "model_type": "ctgan",
    "config": {
      "epochs": 300,
      "batch_size": 500
    }
  }'
```

```
import requests
import time

resp = requests.post(f"{BASE_URL}/api/synthgen/models", headers={
    "Authorization": "Bearer YOUR_API_KEY",
}, json={
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
    "name": "Transactions CTGAN",
    "model_type": "ctgan",
    "config": {"epochs": 300, "batch_size": 500},
})
model_id = resp.json()["model_id"]
print("Model training started:", model_id)

# Poll until ready
while True:
    r = requests.get(f"{BASE_URL}/api/synthgen/models/{model_id}", headers={
        "Authorization": "Bearer YOUR_API_KEY",
    })
    status = r.json()["model"]["status"]
    print(f"  Status: {status}")
    if status in ("ready", "failed"):
        break
    time.sleep(10)
```

Response `201 Created`

```
{
  "id": "5e6f7a8b-9c0d-1234-ef56-789012345678",
  "model_id": "5e6f7a8b-9c0d-1234-ef56-789012345678",
  "job_id": "6f7a8b9c-0d1e-2345-f678-901234567890",
  "status": "pending"
}
```

Model Configuration Reference

Each model type supports specific configuration options passed in the `config` object:

CTGAN (Conditional Tabular GAN)

Option	Type	Default	Description
`epochs`	integer	300	Number of training epochs.
`batch_size`	integer	500	Training batch size.
`embedding_dim`	integer	128	Dimensionality of data embeddings.
`generator_dim`	array[int]	[256, 256]	Hidden layer sizes for the generator network.
`discriminator_dim`	array[int]	[256, 256]	Hidden layer sizes for the discriminator network.
`pac`	integer	10	PAC (Packing) size. Must evenly divide `batch_size`.

CopulaGAN

Same options as CTGAN. Additionally captures complex column correlations using copula functions.

TVAE (Tabular VAE)

Option	Type	Default	Description
`epochs`	integer	300	Number of training epochs.
`batch_size`	integer	500	Training batch size.
`embedding_dim`	integer	128	Dimensionality of data embeddings.

Gaussian Copula

Fastest model type. No deep learning – uses statistical copula fitting. Minimal configuration required:

Option	Type	Default	Description
`seed`	integer	None	Random seed for reproducibility.

Model Type Comparison

Type	Speed	Quality	Best For	Limitations
`ctgan`	Slow	High	Mixed column types, general use	Requires GPU for large datasets
`copulagan`	Slow	High	Complex correlations	Higher memory usage
`tvae`	Medium	Medium-High	Large datasets, fast iteration	May miss complex correlations
`gaussian_copula`	Fast	Medium	Quick prototyping, baselines	Assumes Gaussian distributions

9.9.2 List Models

```
GET /api/synthgen/models
```

Query Parameters

Parameter	Type	Default	Description
`project_id`	string	–	Filter by project.
`limit`	integer	50	Max items.
`offset`	integer	0	Pagination offset.

Example

```
curl "$BASE_URL/api/synthgen/models?project_id=d4e5f6a7-b8c9-0123-def4-567890123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`


```
{
  "items": [
    {
      "id": "5e6f7a8b-9c0d-1234-ef56-789012345678",
      "name": "Transactions CTGAN",
      "model_type": "ctgan",
      "status": "ready",
      "dataset_name": "Transactions Q4",
      "dataset_version": 0,
      "created_at": "2026-02-25T10:00:00Z"
    }
  ],
  "total": 1,
  "limit": 50,
  "offset": 0
}
```

9.9.3 Get Model Detail

```
GET /api/synthgen/models/{model_id}
```

Return model metadata and its associated training/generation jobs.

Example

```
curl "$BASE_URL/api/synthgen/models/5e6f7a8b-9c0d-1234-ef56-789012345678" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "model": {
    "id": "5e6f7a8b-9c0d-1234-ef56-789012345678",
    "name": "Transactions CTGAN",
    "model_type": "ctgan",
    "status": "ready",
    "config": {"epochs": 300, "batch_size": 500},
    "dataset_name": "Transactions Q4",
    "dataset_version": 0,
    "dataset_id": "e5f6a7b8-c9d0-1234-5678-90abcdef1234",
    "created_at": "2026-02-25T10:00:00Z"
  },
  "jobs": [
    {
      "id": "7a8b9c0d-1e2f-3456-7890-123456789012",
      "job_type": "train",
      "status": "completed",
      "created_at": "2026-02-25T10:00:00Z"
    }
  ]
}
```

9.9.4 Generate Synthetic Data

```
POST /api/synthgen/models/{model_id}/generate
```

Generate synthetic data from a trained model. Enqueues a `synthgen_generate` background job. The model must be in `ready` status.

Request Body

Field	Type	Required	Description
`num_rows`	integer	No	Number of synthetic rows to generate (default 1000).
`seed`	integer	No	Random seed for reproducibility.
`conditions`	object	No	Conditional generation constraints (column-value pairs).

Example

```
curl -X POST "$BASE_URL/api/synthgen/models/5e6f7a8b-9c0d-1234-ef56-789012345678/generate
↪" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "num_rows": 10000,
    "seed": 42
  }'
```

```python
# Generate synthetic data
resp = requests.post(
    f"{BASE_URL}/api/synthgen/models/5e6f7a8b-9c0d-1234-ef56-789012345678/generate",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    json={"num_rows": 10000, "seed": 42},
)
synthgen_job_id = resp.json()["synthgen_job_id"]

# Poll until generation completes
while True:
    r = requests.get(
        f"{BASE_URL}/api/synthgen/jobs/{synthgen_job_id}",
        headers={"Authorization": "Bearer YOUR_API_KEY"},
    )
    status = r.json()["status"]
    if status in ("completed", "failed"):
        break
    time.sleep(5)

# Download the generated data
dl = requests.get(
    f"{BASE_URL}/api/synthgen/jobs/{synthgen_job_id}/download",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
with open("synthetic_data.csv", "wb") as f:
    f.write(dl.content)
```

Response `200 OK`

```
{
  "job_id": "8b9c0d1e-2f3a-4567-8901-234567890123",
  "synthgen_job_id": "9c0d1e2f-3a4b-5678-9012-345678901234",
  "status": "pending"
}
```

9.9.5 Delete Model

```
DELETE /api/synthgen/models/{model_id}
```

Permanently delete a SynthGen model and its associated jobs.

Example

```
curl -X DELETE "$BASE_URL/api/synthgen/models/5e6f7a8b-9c0d-1234-ef56-789012345678" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "ok": true,
  "model_id": "5e6f7a8b-9c0d-1234-ef56-789012345678"
}
```

9.9.6 List Jobs

```
GET /api/synthgen/jobs
```

Return SynthGen jobs (training and generation) with optional filters.

Query Parameters

Parameter	Type	Default	Description
`project_id`	string	–	Filter by project.
`model_id`	string	–	Filter by SynthGen model.
`status`	string	–	Filter by status: `pending`, `running`, `completed`, `failed`.
`limit`	integer	50	Max items.
`offset`	integer	0	Pagination offset.

Example

```
curl "$BASE_URL/api/synthgen/jobs?model_id=5e6f7a8b-9c0d-1234-ef56-789012345678" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "9c0d1e2f-3a4b-5678-9012-345678901234",
      "model_id": "5e6f7a8b-9c0d-1234-ef56-789012345678",
      "model_name": "Transactions CTGAN",
      "model_type": "ctgan",
      "job_type": "generate",
      "status": "completed",
      "config": {"num_rows": 10000, "seed": 42},
      "created_at": "2026-02-25T11:00:00Z"
    }
  ],
  "total": 1,
  "limit": 50,
  "offset": 0
}
```

9.9.7 Get Job Detail

```
GET /api/synthgen/jobs/{job_id}
```

Return details for a specific SynthGen job.

Example

```
curl "$BASE_URL/api/synthgen/jobs/9c0d1e2f-3a4b-5678-9012-345678901234" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "id": "9c0d1e2f-3a4b-5678-9012-345678901234",
  "model_id": "5e6f7a8b-9c0d-1234-ef56-789012345678",
  "model_name": "Transactions CTGAN",
  "model_type": "ctgan",
  "project_name": "Fraud Detection v2",
  "job_type": "generate",
  "status": "completed",
  "config": {"num_rows": 10000, "seed": 42},
  "output_artifact_id": "0d1e2f3a-4b5c-6789-0123-456789012345",
  "created_at": "2026-02-25T11:00:00Z"
}
```

9.9.8 Download Synthetic Data

```
GET /api/synthgen/jobs/{job_id}/download
```

Download the output artifact from a completed generation job.

Returns `409 Conflict` if the job output is not available yet.

Example

```
curl -o synthetic_data.csv \
  "$BASE_URL/api/synthgen/jobs/9c0d1e2f-3a4b-5678-9012-345678901234/download" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/synthgen/jobs/9c0d1e2f-3a4b-5678-9012-345678901234/download",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
with open("synthetic_data.csv", "wb") as f:
    f.write(resp.content)
```

Response `200 OK`

Binary file download.

9.9.9 List Model Types

```
GET /api/synthgen/model-types
```

Return available model architectures and their descriptions.

Example

```
curl "$BASE_URL/api/synthgen/model-types" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "model_types": [
    {
      "id": "ctgan",
      "name": "CTGAN",
      "description": "Conditional Tabular GAN - best for general tabular data"
    },
    {
      "id": "copulagan",
      "name": "CopulaGAN",
      "description": "Copula-based GAN - good for capturing correlations"
    },
    {
      "id": "tvae",
      "name": "TVAE",
      "description": "Tabular VAE - faster training, good for large datasets"
    },
    {
      "id": "gaussian_copula",
      "name": "Gaussian Copula",
      "description": "Statistical copula model - fastest, good baseline"
    }
  ]
}
```

9.9.10 Full Workflow Example

Train a CTGAN model and generate 10,000 synthetic rows:

```
import requests
import time

BASE_URL = "http://localhost:8888"
HEADERS = {"Authorization": "Bearer YOUR_API_KEY"}

# 1. Train a CTGAN model
resp = requests.post(f"{BASE_URL}/api/synthgen/models", headers=HEADERS, json={
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
    "name": "Transactions CTGAN v1",
```

(continues on next page)

(continued from previous page)

```
    "model_type": "ctgan",
    "config": {"epochs": 300},
})
model_id = resp.json()["model_id"]

# 2. Wait for training to complete
while True:
    r = requests.get(f"{BASE_URL}/api/synthgen/models/{model_id}", headers=HEADERS)
    if r.json()["model"]["status"] in ("ready", "failed"):
        break
    time.sleep(15)

# 3. Generate synthetic data
resp = requests.post(
    f"{BASE_URL}/api/synthgen/models/{model_id}/generate",
    headers=HEADERS,
    json={"num_rows": 10000, "seed": 42},
)
job_id = resp.json()["synthgen_job_id"]

# 4. Wait for generation to complete
while True:
    r = requests.get(f"{BASE_URL}/api/synthgen/jobs/{job_id}", headers=HEADERS)
    if r.json()["status"] in ("completed", "failed"):
        break
    time.sleep(5)

# 5. Download the result
resp = requests.get(
    f"{BASE_URL}/api/synthgen/jobs/{job_id}/download",
    headers=HEADERS,
)
with open("synthetic_transactions.csv", "wb") as f:
    f.write(resp.content)
print(f"Generated {10000} synthetic rows.")
```

See also

- *Datasets API* – Uploading real datasets for training.
- *Reports API* – Generating SynthGen quality reports.

9.10 ML Studio (What-If) API

ML Studio enables what-if analysis against deployed models. Create a session with a baseline input, define scenarios that change specific features, run predictions for each scenario, and compare the results.

All endpoints are prefixed with `/api/studio`.

Endpoints

- *Create Session*
- *List Sessions*
- *Get Session Detail*
- *Get Feature Schema*
- *Create Scenario*
- *Run Scenario*
- *Compare Scenarios*
- *End-to-End What-If Example*

9.10.1 Create Session

```
POST /api/studio/sessions
```

Create a new what-if analysis session. The baseline input is immediately sent for prediction to establish the reference point. The session stores the baseline prediction, feature contributions (SHAP values), and the feature schema derived from the underlying dataset.

Request Body

Field	Type	Required	Description
`project_id`	string	Yes	UUID of the project.
`deployment_id`	string	Yes	UUID of the active deployment to analyze.
`baseline_input`	object	Yes	Feature values for the baseline prediction. Only features from the model's schema are used; the target column is automatically excluded.

Example

```
curl -X POST "$BASE_URL/api/studio/sessions" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
    "baseline_input": {
      "amount": 29.99,
      "merchant_id": 42,
      "hour": 14,
      "day_of_week": 3
    }
  }'
```

```python
import requests

resp = requests.post(f"{BASE_URL}/api/studio/sessions", headers={
    "Authorization": "Bearer YOUR_API_KEY",
}, json={
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
    "baseline_input": {
        "amount": 29.99,
        "merchant_id": 42,
        "hour": 14,
        "day_of_week": 3,
    },
})
session_id = resp.json()["studio_session_id"]
print("Session created:", session_id)
```

Response `201 Created`

```
{
  "id": "1e2f3a4b-5c6d-7890-1234-567890abcdef",
  "studio_session_id": "1e2f3a4b-5c6d-7890-1234-567890abcdef"
}
```

9.10.2 List Sessions

```
GET /api/studio/sessions
```

Query Parameters

Parameter	Type	Default	Description
`project_id`	string	–	**Required.** Filter by project.
`limit`	integer	50	Max items (1–100).
`offset`	integer	0	Pagination offset.

Example

```
curl "$BASE_URL/api/studio/sessions?project_id=d4e5f6a7-b8c9-0123-def4-567890123456" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "items": [
    {
      "id": "1e2f3a4b-5c6d-7890-1234-567890abcdef",
      "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
      "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
      "created_at": "2026-02-28T09:00:00Z"
    }
```

(continues on next page)

(continued from previous page)

```
  ],
  "total": 1,
  "limit": 50,
  "offset": 0
}
```

9.10.3 Get Session Detail

```
GET /api/studio/sessions/{session_id}
```

Return the session with baseline input/output, feature schema, and all scenarios.

Example

```
curl "$BASE_URL/api/studio/sessions/1e2f3a4b-5c6d-7890-1234-567890abcdef" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/studio/sessions/1e2f3a4b-5c6d-7890-1234-567890abcdef",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
session = resp.json()
print("Baseline prediction:", session["baseline_output"])
for s in session["scenarios"]:
    print(f"  Scenario '{s['name']}': {s['output']}")
```

Response `200 OK`

```
{
  "id": "1e2f3a4b-5c6d-7890-1234-567890abcdef",
  "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
  "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
  "feature_schema": {
    "amount": "float64",
    "merchant_id": "int64",
    "hour": "int64",
    "day_of_week": "int64"
  },
  "baseline_input": {
    "amount": 29.99,
    "merchant_id": 42,
    "hour": 14,
    "day_of_week": 3
  },
  "baseline_output": {
    "prediction": "0"
  },
  "baseline_contributions": [
    {"feature": "amount", "value": -0.32},
    {"feature": "merchant_id", "value": -0.15},
```

(continues on next page)

(continued from previous page)

```
    {"feature": "hour", "value": -0.08},
    {"feature": "BiasTerm", "value": -1.20}
  ],
  "created_at": "2026-02-28T09:00:00Z",
  "scenarios": [
    {
      "id": "2f3a4b5c-6d7e-8901-2345-67890abcdef1",
      "name": "High amount at night",
      "description": "What if amount is 10x and hour is 3am?",
      "changes": {"amount": 9500.00, "hour": 3},
      "status": "succeeded",
      "scenario_input": {
        "amount": 9500.00,
        "merchant_id": 42,
        "hour": 3,
        "day_of_week": 3
      },
      "output": {"prediction": "1"},
      "contributions": [
        {"feature": "amount", "value": 1.85},
        {"feature": "hour", "value": 0.92},
        {"feature": "BiasTerm", "value": -1.20}
      ],
      "error": null,
      "created_at": "2026-02-28T09:05:00Z"
    }
  ]
}
```

9.10.4 Get Feature Schema

```
GET /api/studio/deployments/{deployment_id}/schema
```

Return the feature schema for a deployment, derived from the training dataset's column metadata. The target column is automatically excluded.

Example

```
curl "$BASE_URL/api/studio/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/schema" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

Response `200 OK`

```
{
  "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
  "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
  "model_id": "a7b8c9d0-e1f2-3456-7890-abcdef123456",
  "experiment_id": "b8c9d0e1-f2a3-4567-8901-bcdef1234567",
  "dataset_version_id": "f6a7b8c9-d0e1-2345-6789-0abcdef12345",
  "target_column": "is_fraud",
  "feature_schema": {
```

(continues on next page)

(continued from previous page)

```
    "amount": "float64",
    "merchant_id": "int64",
    "hour": "int64",
    "day_of_week": "int64"
  }
}
```

9.10.5 Create Scenario

```
POST /api/studio/sessions/{session_id}/scenarios
```

Define a what-if scenario by specifying which features to change relative to the baseline input. Only the `changes` dict is required; unchanged features are inherited from the baseline.

Request Body

Field	Type	Required	Description
`name`	string	Yes	Scenario name.
`description`	string	No	Optional description of the hypothesis.
`changes`	object	Yes	Dictionary of feature-value overrides. Keys must match the feature schema.

Example

```
curl -X POST "$BASE_URL/api/studio/sessions/1e2f3a4b-5c6d-7890-1234-567890abcdef/
↪scenarios" \
  -H "Authorization: Bearer YOUR_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "name": "High amount at night",
    "description": "What if the transaction amount is 10x higher and happens at 3am?",
    "changes": {
      "amount": 9500.00,
      "hour": 3
    }
  }'
```

```
resp = requests.post(
    f"{BASE_URL}/api/studio/sessions/1e2f3a4b-5c6d-7890-1234-567890abcdef/scenarios",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
    json={
        "name": "High amount at night",
        "description": "What if amount is 10x and hour is 3am?",
        "changes": {"amount": 9500.00, "hour": 3},
    },
)
scenario_id = resp.json()["scenario_id"]
```

Response `201 Created`

```
{
  "id": "2f3a4b5c-6d7e-8901-2345-67890abcdef1",
  "scenario_id": "2f3a4b5c-6d7e-8901-2345-67890abcdef1",
  "status": "pending"
}
```

9.10.6 Run Scenario

```
POST /api/studio/scenarios/{scenario_id}/run
```

Execute the scenario by applying the feature changes to the baseline input and running a prediction through the deployment. The scenario transitions from `pending` to `running` to `succeeded` (or `failed`).

Example

```
curl -X POST "$BASE_URL/api/studio/scenarios/2f3a4b5c-6d7e-8901-2345-67890abcdef1/run" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.post(
    f"{BASE_URL}/api/studio/scenarios/2f3a4b5c-6d7e-8901-2345-67890abcdef1/run",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
result = resp.json()
print("Scenario prediction:", result["output"]["prediction"])
```

Response `200 OK`

```
{
  "ok": true,
  "output": {
    "prediction": "1"
  },
  "contributions": [
    {"feature": "amount", "value": 1.85},
    {"feature": "hour", "value": 0.92},
    {"feature": "merchant_id", "value": -0.15},
    {"feature": "BiasTerm", "value": -1.20}
  ]
}
```

9.10.7 Compare Scenarios

```
GET /api/studio/sessions/{session_id}/compare
```

Compare the baseline with all completed scenarios. Returns the baseline output and, for each scenario, the output and a delta (difference in prediction value and top contributing features).

Example

```
curl "$BASE_URL/api/studio/sessions/1e2f3a4b-5c6d-7890-1234-567890abcdef/compare" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

```
resp = requests.get(
    f"{BASE_URL}/api/studio/sessions/1e2f3a4b-5c6d-7890-1234-567890abcdef/compare",
    headers={"Authorization": "Bearer YOUR_API_KEY"},
)
data = resp.json()
print("Baseline:", data["baseline"]["output"])
for s in data["scenarios"]:
    print(f"  {s['name']}: prediction delta = {s['delta'].get('prediction')}")
```

Response `200 OK`


```
{
  "baseline": {
    "output": {"prediction": "0"},
    "contributions": [
      {"feature": "amount", "value": -0.32},
      {"feature": "merchant_id", "value": -0.15},
      {"feature": "BiasTerm", "value": -1.20}
    ]
  },
  "scenarios": [
    {
      "id": "2f3a4b5c-6d7e-8901-2345-67890abcdef1",
      "name": "High amount at night",
      "output": {"prediction": "1"},
      "delta": {
        "prediction": null,
        "contributions": [
          {"feature": "amount", "delta": 2.17},
          {"feature": "hour", "delta": 1.00},
          {"feature": "merchant_id", "delta": 0.0}
        ]
      }
    }
  ]
}
```


Note

When the prediction is categorical (classification), the `prediction` delta is `null` because difference between class labels is not meaningful. For regression models, the delta is the numeric difference between the scenario and baseline predictions.

Contribution deltas are always numeric and represent the change in SHAP value for each feature. They are sorted by absolute delta (descending) and limited to the top 10 features.

9.10.8 End-to-End What-If Example

```python
import requests

BASE_URL = "http://localhost:8888"
HEADERS = {"Authorization": "Bearer YOUR_API_KEY"}

# 1. Get the feature schema for a deployment
schema = requests.get(
    f"{BASE_URL}/api/studio/deployments/d0e1f2a3-b4c5-6789-0123-def456789012/schema",
    headers=HEADERS,
).json()
print("Features:", list(schema["feature_schema"].keys()))

# 2. Create a session with baseline input
resp = requests.post(f"{BASE_URL}/api/studio/sessions", headers=HEADERS, json={
    "project_id": "d4e5f6a7-b8c9-0123-def4-567890123456",
    "deployment_id": "d0e1f2a3-b4c5-6789-0123-def456789012",
    "baseline_input": {
        "amount": 29.99,
        "merchant_id": 42,
        "hour": 14,
        "day_of_week": 3,
    },
})
session_id = resp.json()["studio_session_id"]

# 3. Create scenarios
scenarios = [
    {"name": "High amount", "changes": {"amount": 9500.00}},
    {"name": "Late night", "changes": {"hour": 3}},
    {"name": "High amount + late night", "changes": {"amount": 9500.00, "hour": 3}},
]
scenario_ids = []
for s in scenarios:
    r = requests.post(
        f"{BASE_URL}/api/studio/sessions/{session_id}/scenarios",
        headers=HEADERS,
        json={"name": s["name"], "changes": s["changes"]},
    )
    scenario_ids.append(r.json()["scenario_id"])

# 4. Run all scenarios
for sid in scenario_ids:
    requests.post(f"{BASE_URL}/api/studio/scenarios/{sid}/run", headers=HEADERS)

# 5. Compare results
comparison = requests.get(
    f"{BASE_URL}/api/studio/sessions/{session_id}/compare",
    headers=HEADERS,
).json()

print(f"Baseline: {comparison['baseline']['output']}")
```

(continues on next page)

(continued from previous page)

```
for s in comparison["scenarios"]:
    print(f"  {s['name']}: {s['output']} (delta={s['delta']})")
```

See also

- *Deployments API* – Creating and managing deployments.
- *Models API* – Model details and direct predictions.

9.11 Error Handling

All error responses use a consistent JSON format. This page documents the HTTP status codes, error body format, and common error scenarios for each module.

On this page

- *Error Response Format*
- *HTTP Status Codes*
- *Rate Limiting*
- *Common Error Scenarios*
- *Retry Strategy*

9.11.1 Error Response Format

All errors return a JSON body with a `detail` field:

```
{
  "detail": "Human-readable error message"
}
```

Some validation errors (HTTP 422) from Pydantic return a structured list of field-level errors:

```
{
  "detail": [
    {
      "loc": ["body", "email"],
      "msg": "value is not a valid email address",
      "type": "value_error.email"
    }
  ]
}
```

9.11.2 HTTP Status Codes

Code	Name	Description
200	OK	Request succeeded. The response body contains the result.
201	Created	Resource was successfully created. The response body contains the new resource ID.
302	Found	Redirect to another URL (used by download endpoints).
400	Bad Request	The request body or parameters are malformed. Check the `detail` message for specifics.
401	Unauthorized	No valid authentication credentials provided. Either the session has expired or the API key is missing/invalid.
403	Forbidden	The authenticated user does not have permission for this action. This can occur when accessing another user's resources or when the API key lacks the required scope.
404	Not Found	The requested resource does not exist, or the authenticated user does not have access to it. For privacy, the server does not distinguish between "does not exist" and "no access."
409	Conflict	The request conflicts with the current state. Common causes: report artifact not ready yet, duplicate resource name, or resource is still processing.
422	Unprocessable Entity	Request body failed validation. The `detail` field contains the specific validation error. Common causes: missing required field, invalid enum value, empty project name, invalid model_type.
429	Too Many Requests	Rate limit exceeded. Wait for the number of seconds indicated in the `Retry-After` header before retrying.
500	Internal Server Error	An unexpected error occurred. The `detail` field may contain information about the failure. If this persists, check server logs.

9.11.3 Rate Limiting

Rate-limited endpoints include the following headers in every response:

Header	Description
`X-RateLimit-Limit`	Maximum number of requests allowed in the current window.
`X-RateLimit-Remaining`	Number of requests remaining in the current window.
`X-RateLimit-Reset`	Unix timestamp when the rate limit window resets.
`Retry-After`	(Only on 429 responses) Seconds to wait before retrying.

Rate-limited endpoints:

- **Authentication** – `POST /api/auth/login`, `POST /api/auth/register`, `POST /api/auth/forgot-password`
- **Predictions** – `POST /api/mlops/deployments/{id}/predict`, `POST /api/models/{id}/predict`

9.11.4 Common Error Scenarios

Authentication Errors

Code	Error	Cause
401	`Missing or invalid credentials`	No session cookie or API key, or the credentials have expired.
401	`Invalid email or password`	Wrong password during login.
403	`Insufficient scope`	API key does not have the required scope (e.g. `predict` scope missing for prediction endpoints).
409	`User already exists`	Attempting to register with an email that is already taken.
422	`Password must be at least 8 characters`	Password too short during registration or reset.
429	`Rate limit exceeded`	Too many login/register attempts.

Project Errors

Code	Error	Cause
404	`Project not found`	Project ID does not exist or the user is not a member.
422	`Project name must not be empty`	Name was blank or whitespace-only.

Dataset Errors

Code	Error	Cause
404	`Dataset not found`	Dataset ID does not exist or belongs to another project.
404	`Dataset version not found`	No version exists for the given dataset.
422	`Unsupported file type`	Uploaded file is not CSV or Parquet.

Experiment Errors

Code	Error	Cause
404	`Experiment not found`	Experiment ID does not exist or is inaccessible.
422	`target_column not found in dataset`	The specified target column does not exist in the dataset schema.
422	`Invalid problem_type`	Must be `classification` or `regression`.

Model Errors

Code	Error	Cause
404	`Model not found`	Model ID does not exist or is inaccessible.
500	`Prediction failed`	H2O model failed to produce a prediction. Check input features match the training schema.

Deployment Errors

Code	Error	Cause
404	`Deployment not found or inactive`	Deployment does not exist or has been deactivated.
422	`stage must be 'staging' or 'production'`	Invalid stage value when creating a deployment.
422	`model_id must belong to project`	The model is from a different project.
422	`to_stage must be 'production'`	Invalid promote target.
422	`Provide to_deployment_id or to_model_id`	Rollback request missing both identifiers.

Report Errors

Code	Error	Cause
409	`Report has no artifact yet (still processing)`	Attempting to view or download a report that is still being generated.
422	`Invalid kind`	Report kind must be one of: `deployment`, `experiment`, `model`, `project`, `synthgen`.

Privacy Suite Errors

Code	Error	Cause
404	`Session not found`	Privacy session does not exist or is inaccessible.
404	`Anonymized output not found`	Attempting to download before transformation is complete.
404	`Rule not found`	Rule ID does not exist in the given policy.
422	`dataset_id or dataset_version_id is required`	Session creation without specifying which dataset to scan.
422	`Dataset must belong to the same project as the policy`	Cross-project dataset/policy mismatch.

SynthGen Errors

Code	Error	Cause
400	`Model is not ready`	Attempting to generate from a model that is still training.
404	`Model not found`	SynthGen model does not exist or is inaccessible.
409	`Job output not available yet`	Attempting to download before generation completes.
422	`Invalid model_type`	Model type must be one of: `ctgan`, `copulagan`, `tvae`, `gaussian_copula`.
422	`num_rows must be between 1 and N`	Requested row count exceeds the streaming limit.

Studio Errors

Code	Error	Cause
404	`Deployment not found or inactive`	The deployment linked to the session is no longer active.
500	`Baseline prediction failed`	The baseline input caused a prediction error. Check that input features match the deployment's feature schema.
500	`Scenario prediction failed`	A scenario's modified input caused a prediction error.

9.11.5 Retry Strategy

For transient errors (429, 500, 502, 503, 504) the recommended retry strategy is exponential backoff:

```python
import time
import requests

def api_request(method, url, **kwargs):
    """Make an API request with exponential backoff retry."""
    max_retries = 3
    for attempt in range(max_retries + 1):
        resp = requests.request(method, url, **kwargs)

        if resp.status_code == 429:
            wait = int(resp.headers.get("Retry-After", 2 ** attempt))
            time.sleep(wait)
            continue

        if resp.status_code >= 500 and attempt < max_retries:
            time.sleep(2 ** attempt)
            continue

        resp.raise_for_status()
        return resp.json()

    raise Exception(f"Request failed after {max_retries} retries")
```

See also

- *Authentication* – How to authenticate API requests.
- *API Reference* – API overview and conventions.