# How to handle API Errors

Last modified on 08/18/2026 2:16 pm EDT

In this article, we will discuss the various API errors that you may encounter. The goal here is to provide awareness of our error types and provide a few ways you can handle them.

## 4XX Level Errors

| |
|---|
| 400 - Bad Request |
| 401 - Unauthorized |
| 403 - Forbidden (Permission Denied) |
| 404 - Not Found |
| 405 - Method Not Allowed |
| 409 - Conflict (Scheduled time overlaps with an existing appointment; "Appointment overlaps with [Appointment ID]") |
| 429 - Too Many Requests ("detail": "Request was throttled. Expected available in X seconds.") |

## 400 - Bad Request

400 API errors often arise from client-side mistakes, typically indicating that the server cannot process the request due to invalid syntax or parameters. The best way to handle this error is to do the following.

1. Use logging frameworks to log errors and other relevant information.

2. Use specific error handling in your code.
   -> Try-catch blocks
   -> async/await
   -> Or any other similar tactics

3. Use Postman or other services that provide API testing.

## 401 - Unauthorized

401 occurs when the access token has expired or is invalidated. Here are some scenarios where you may encounter this issue.
[To note: our refresh token does not expire.]

1. The access token has reached its lifetime (access token expiration is 48 hours)

2. Going through the authentication process with the same DrChrono user, this will invalidate the initial refresh and access tokens.

3. Refreshing the access token while using the prior access token.

4. When refreshing a token but encounter a 401 - "error": "invalid_grant" or "invalid_client".
   -> Refresh Token may be invalid: This may be caused by going through the OAuth process again or thetoken is revoked.
   -> Incorrect refresh token
   -> The client ID or Secret is incorrect
   -> grant_type is incorrect

The best way to handle these errors is to ensure proper token handling when generating a new token. Ensure only one job runs the refresh execution, and be aware of how the API tokens/keys are handled.

**If you are encountering premature or erratic 401s, please provide the following details and report them to api@drchrono.com**

1. When did you first notice the issue?
2. Can you provide a brief description of how you generate a new access token?
3. How are your tokens handled? Is there a cache system/etc?
4. Can you provide your code's environment? (Python/JS/etc)
5. If web-based, what browser is being used?
6. Who is the authenticated user? Please provide a username or user_id

## 403 - Permission Error

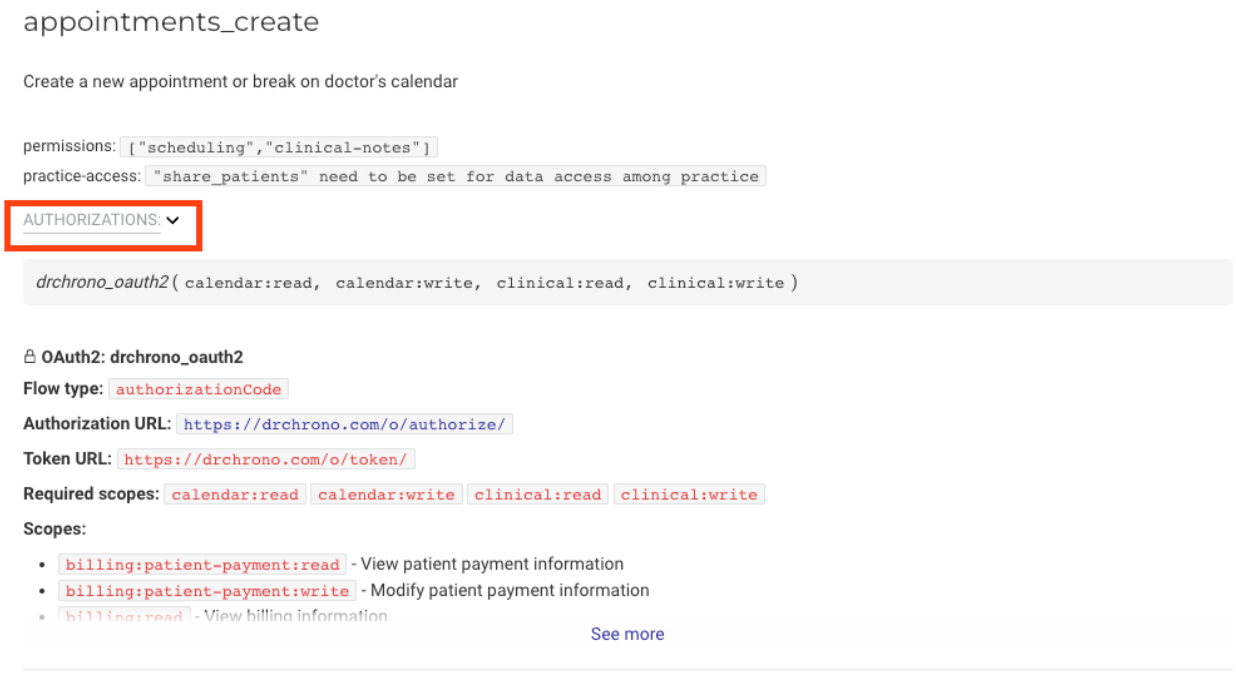
A 403 HTTP error code shows that the current token user does not have the correct permissions or scopes to access certain APIs.

#### User Permissions

To check your user permissions, navigate to the Account tab > Staff Permissions. From the list, select your API token user and edit permissions. Enable the specific permissions needed for the API endpoint being used. This information can be found in our API Documentation > "permissions" and "practice-access".

#### Scopes

The correct scope is shown within the "AUTHORIZATIONS" tab of our documentation.




The token user can be viewed from the Users API:

```
GET https://app.drchrono.com/api/users/current
```

## 405 - Method Not Allowed

405 HTTP code usually occurs when you are using the incorrect action.

EXAMPLE:

```
PATCH https://app.drchrono.com/api/patients?since=2024-06-11
```

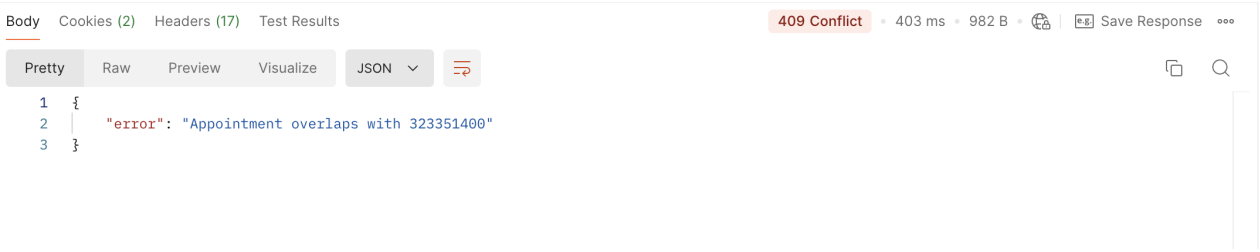
RESPONSE:

```
{
"detail": "Method \"PATCH\" not allowed."
}
```

## 409 - Appointment Conflict

When creating an appointment that overlaps an existing appointment, you will encounter a 409 error. If your practice would like to schedule overlapping appointments, you can enable an account setting to allow this action.

To enable appointment overlap, navigate to the Account Settings page > General > Calendar Settings. Enable the "Allow Exam Room Overlaps".




## 429 - Too Many Requests

There are a few levels of throttling. We will go through each one in this section.

**Application-level rate limit**. Each API application has a default rate limit of 500 requests per hour. The limit is always reset at the top of the hour.

To avoid hitting the limit,
-> Understand your expected API usage and manage your code appropriately. Please ensure you do not use redundant calls and distribute requests over time.
-> We offer a list of bulk API resources.

If you hit the limit, ensure you have the appropriate measures in your code, including exponential backoff, or create custom code to retry once the limit is reset. You can request a rate limit increase for your DrChrono API app.

**Email api@drchrono.com and provide the following information.**
-> The name of your API application
-> Context regarding your API calls/methods
-> Expected API usage

# 5XX Level Errors

| |
|---|
| 500 - Internal Server Error |
| 502 - Bad Gateway |
| 503 - Service Temporarily Unavailable |
| 504 - Gateway Timeout |

When any internal server errors occur, include a 25-30 second retry logic once it detects the above HTTP error code. When you retry 2 or 3 more times with an exponential backoff period, oftentimes you can get a successful call. Incorporating jitter randomization can also be useful.

We understand that this is not the best experience, but it should be effective and useful in high-urgency situations.

**Once detected, please report this instance to api@drchrono.com. In your email, please provide the following information.**

1. The name of the API application in use. This can be found on the DrChrono webpage > Account > API.
2. The complete API endpoint URL. Example: GET https://app.drchrono.com/api/patients?since=2016-01-01

   If you are incurring 5XX HTTP errors for any other action (POST/PATCH/etc) please provide an example of

the request payload sent. If there is any PHI information, redact the info like: *"Field": REDACT*.

3. The connected API user. You can call GET /api/users/current to obtain the token user.

4. When the issue first started and the timestamps of when they occurred. (Please provide the timezone)

5. Provide any other helpful information
   -> What is your use case for utilizing that specific endpoint?
   -> When does your system perform its task?