API Error Codes Reference

This document describes all error codes used in the CareLinkAI API.

Error Response Format

All API errors follow this standardized format:

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Human-readable error message",
    "details": {
      "field": "fieldName",
      "constraint": "unique"
    }
  }
}
```

Error Codes

Authentication & Authorization

`UNAUTHORIZED` (401)

Description: The request requires authentication but no valid credentials were provided.

Common Causes:
- No session cookie/token
- Expired session
- Invalid credentials

Client Action: Redirect to login page

Example:

```
{
  "success": false,
  "error": {
    "code": "UNAUTHORIZED",
    "message": "Authentication required"
  }
}
```

`FORBIDDEN` (403)

Description: The authenticated user does not have permission to access this resource.

Common Causes:
- Wrong user role
- Not resource owner
- Attempting to access another user's data

Client Action: Show "Access denied" message

Example:

```
{
  "success": false,
  "error": {
    "code": "FORBIDDEN",
    "message": "You do not have permission to access this resource"
  }
}
```

`INVALID_CREDENTIALS` (401)

Description: The provided credentials are incorrect.

Common Causes:
- Wrong email/password combination
- Account locked or disabled

Client Action: Show error near login form

Example:

```
{
  "success": false,
  "error": {
    "code": "INVALID_CREDENTIALS",
    "message": "Invalid email or password"
  }
}
```

`SESSION_EXPIRED` (401)

Description: The user's session has expired.

Client Action: Redirect to login with "session expired" message

---

Validation Errors

`VALIDATION_ERROR` (400)

Description: The request body or parameters failed validation.

Common Causes:
- Missing required fields

- Invalid field format
- Values out of range
- Type mismatch

Client Action: Display validation errors next to form fields

Example:

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid input",
    "details": {
      "errors": {
        "email": {
          "_errors": ["Invalid email address"]
        },
        "password": {
          "_errors": ["Password must be at least 8 characters"]
        }
      }
    }
  }
}
```

`INVALID_INPUT` (400)

Description: Generic validation error for single fields.

Example:

```
{
  "success": false,
  "error": {
    "code": "INVALID_INPUT",
    "message": "File type not supported",
    "details": {
      "field": "file",
      "allowedTypes": ["image/jpeg", "image/png", "application/pdf"]
    }
  }
}
```

`MISSING_REQUIRED_FIELD` (400)

Description: A required field is missing from the request.

Example:

```
{
  "success": false,
  "error": {
    "code": "MISSING_REQUIRED_FIELD",
    "message": "Email is required",
    "details": {
      "field": "email"
    }
  }
}
```

Resource Errors

`NOT_FOUND` (404)

Description: The requested resource does not exist.

Common Causes:
- Invalid ID in URL
- Resource was deleted
- User doesn't have access

Client Action: Show "Not found" page or message

Example:

```
{
  "success": false,
  "error": {
    "code": "NOT_FOUND",
    "message": "User not found"
  }
}
```

`ALREADY_EXISTS` (409)

Description: A resource with this identifier already exists.

Common Causes:
- Duplicate email registration
- Unique constraint violation

Client Action: Show "already exists" error near relevant field

Example:

```
{
  "success": false,
  "error": {
    "code": "ALREADY_EXISTS",
    "message": "A user with this email already exists",
    "details": {
      "constraint": "email"
    }
  }
}
```

Operation Errors

`OPERATION_FAILED` (500)

Description: The operation could not be completed.

Common Causes:
- Business logic failure
- State transition not allowed

Example:

```
{
  "success": false,
  "error": {
    "code": "OPERATION_FAILED",
    "message": "Cannot delete account with active bookings"
  }
}
```

`DATABASE_ERROR` (500)

Description: A database operation failed.

Common Causes:
- Connection timeout
- Query failure
- Constraint violation

Client Action: Show generic error, retry button

Example:

```
{
  "success": false,
  "error": {
    "code": "DATABASE_ERROR",
    "message": "A database error occurred"
  }
}
```

`EXTERNAL_SERVICE_ERROR` (502/503)

Description: An external service (S3, payment processor, etc.) is unavailable.

Client Action: Show “service temporarily unavailable” message

Example:

```
{
  "success": false,
  "error": {
    "code": "EXTERNAL_SERVICE_ERROR",
    "message": "File upload service is temporarily unavailable"
  }
}
```

Rate Limiting

`RATE_LIMIT_EXCEEDED` (429)

Description: Too many requests from this client.

Client Action: Show “too many requests” message with retry time

Example:

```
{
  "success": false,
  "error": {
    "code": "RATE_LIMIT_EXCEEDED",
    "message": "Too many requests, please try again later",
    "details": {
      "retryAfter": 60
    }
  }
}
```

Server Errors

`INTERNAL_ERROR` (500)

Description: An unexpected error occurred on the server.

Common Causes:
- Unhandled exception
- Programming error
- System failure

Client Action: Show generic error message, provide support contact

Example:

```
{
  "success": false,
  "error": {
    "code": "INTERNAL_ERROR",
    "message": "An unexpected error occurred"
  }
}
```

HTTP Status Code Mapping

Status	Codes
400	`VALIDATION_ERROR`, `INVALID_INPUT`, `MISSING_REQUIRED_FIELD`
401	`UNAUTHORIZED`, `INVALID_CREDENTIALS`, `SESSION_EXPIRED`
403	`FORBIDDEN`
404	`NOT_FOUND`
409	`ALREADY_EXISTS`
429	`RATE_LIMIT_EXCEEDED`
500	`INTERNAL_ERROR`, `OPERATION_FAILED`, `DATABASE_ERROR`
502/503	`EXTERNAL_SERVICE_ERROR`

Client-Side Error Handling

Example: Generic Error Handler

```
import { showErrorToast, getErrorMessage } from '@/lib/errors/client-errors';

async function handleRequest() {
  try {
    const response = await fetch('/api/resource');
    const data = await handleApiResponse(response);
    // Handle success
  } catch (error) {
    // Automatically extracts and displays error message
    showErrorToast(error);
  }
}
```

Example: Specific Error Handling

```
try {
  const response = await fetch('/api/auth/login', {
    method: 'POST',
    body: JSON.stringify({ email, password }),
  });

  if (!response.ok) {
    const errorData = await response.json();

    switch (errorData.error.code) {
      case 'INVALID_CREDENTIALS':
        setError('Invalid email or password');
        break;
      case 'RATE_LIMIT_EXCEEDED':
        setError('Too many login attempts. Try again later.');
        break;
      default:
        setError('Login failed. Please try again.');
    }
    return;
  }

  // Handle success
} catch (error) {
  setError('Network error. Please check your connection.');
}
```

Server-Side Error Handling

Example: Using Error Utilities

```
import { handleApiError, createErrorResponse, ErrorCode } from '@/lib/errors/api-errors';

export async function POST(req: NextRequest) {
  try {
    // Validation
    if (!body.email) {
      return createErrorResponse(
        ErrorCode.MISSING_REQUIRED_FIELD,
        'Email is required',
        400,
        { field: 'email' }
      );
    }

    // Business logic
    const user = await prisma.user.create({ data: body });

    return NextResponse.json({ success: true, data: user });

  } catch (error) {
    // Automatically handles Prisma errors, generic errors, etc.
    return handleApiError(error, {
      path: req.url,
      method: 'POST',
    });
  }
}
```