# **HTTP Status Codes Documentation**

**Document Version**: 1.0  
 **Last Updated**: February 2, 2025  
 **Subject**: Client (4xx) and Server (5xx) Error Codes

## **Introduction**

This document provides a comprehensive overview of HTTP status codes, focusing on client-side (4xx) and server-side (5xx) error responses. These status codes are essential for understanding and debugging web application issues.

## **Client Error Responses (4xx Series)**

Client error responses indicate that there is an issue with the request sent by the client.

### **400 Bad Request**

**Definition**: The server cannot process the request due to a client error. **Usage Context**: When the request syntax is malformed or invalid. **Common Scenarios**: Invalid request format, missing required parameters, or nonexistent resources.

### **401 Unauthorized**

**Definition**: Authorization is required to access the requested resource. **Technical Details**: The server will specify supported authentication method(s). **Implementation**: Client must authenticate before access is granted. **Note**: Multiple authentication schemes may be supported.

### **403 Forbidden**

**Definition**: The server refuses to fulfill the request despite valid authentication. **Usage Context**: When permissions are insufficient for the requested resource. **Key Distinction**: Unlike 401, authentication is not the issue; rather, it's a permissions problem.

### **404 Not Found**

**Definition**: The requested resource could not be found on the server. **Common Causes**:

* Dead or broken links
* Misspelled URLs
* Removed or relocated resources

### **413 Request Entity Too Large**

**Definition**: The request payload exceeds server limits. **Technical Impact**: Server may close the connection. **Additional Information**: May include a Retry-After header for temporary conditions. **Usage**: Commonly seen with file upload limitations.

### **414 Request URI Too Large**

**Definition**: The URI length exceeds server processing limits. **Technical Context**: Server has predetermined maximum URI length restrictions. **Implementation Note**: Indicates that the request URI is too large for server processing.

### **429 Too Many Requests**

**Definition**: Client has exceeded rate limits. **Technical Details**: Resource unavailable due to request frequency limitations. **Scope**: May apply to specific users or resources. **Implementation**: Typically includes rate limit reset information.

## **Server Error Responses (5xx Series)**

Server error responses indicate that the server failed to fulfill a valid request.

### **500 Internal Server Error**

**Definition**: Generic server error response. **Technical Context**: Server encountered an unexpected condition. **Usage**: Catch-all error when no more specific 5xx error is applicable. **Impact**: Request cannot be fulfilled due to server-side issues.

### **502 Bad Gateway**

**Definition**: Invalid response received from an upstream server. **Common Scenario**: Proxy server (e.g., Nginx) unable to get valid response. **Technical Context**: Often indicates configuration issues or service unavailability. **Usage**: Common in reverse proxy setups.

### **503 Service Unavailable**

**Definition**: Server temporarily unable to handle requests. **Common Causes**:

* Scheduled maintenance
* Server overload
* System outages **Nature**: Temporary condition **Additional Information**: May include expected recovery time.

## **Best Practices for Implementation**

### **Error Handling Guidelines**

1. Implement appropriate error messages for each status code
2. Include relevant debugging information where appropriate
3. Maintain consistent error response formats
4. Log errors for monitoring and analysis

### **Monitoring Recommendations**

1. Track frequency of different error codes
2. Set up alerts for unusual error patterns
3. Monitor response times and error rates
4. Implement proper logging for troubleshooting

## **Conclusion**

Understanding and properly implementing HTTP status codes is crucial for maintaining robust web applications. Regular monitoring and appropriate error handling ensure optimal service delivery and user experience.