LOGZILLA DOCUMENTATION

# Core Concepts

LogZilla API fundamentals covering HTTP methods, JSON conventions, 2xx and 4xx status codes, pagination, and error handling for REST integrations

LogZilla API · Generated September 12, 2026 · logzilla.ai/docs/logzilla-api/core-concepts

# Core Concepts

Essential API concepts for working with LogZilla's RESTful interface, including HTTP conventions, error handling, pagination mechanisms, and data structures. Understanding these fundamentals enables effective API integration and troubleshooting.

# Prerequisites

- Administrative access to LogZilla web interface
- Understanding of REST API concepts and HTTP methods
- JSON data format familiarity
- HTTP client tools (curl, programming language libraries)
- Valid API authentication token (see Getting Started (https://www.logzilla.ai/docs/logzilla-api/getting-started))

# HTTP Methods and Conventions

Requests to the API are made using standard HTTP methods. For endpoints that accept or return data in the request/response body, JSON is the standard format, and the user should typically set the `Content-Type: application/json` header for `POST`, `PUT`, and `PATCH` requests.

# Error Handling

The LogZilla API uses standard HTTP status codes to indicate the success or failure of an API request.

## 2xx (Successful)

- `200 OK`: The request was successful.
- `201 Created`: The request was successful, and a resource was created.
- `202 Accepted`: The request has been accepted for processing, but the processing has not been completed (common for asynchronous operations such as queries). The response body usually contains information on checking the status.
- `204 No Content`: The request was successful, but there is no content to return (e.g., for a successful DELETE request).

## 3xx (Redirection)

- `301 Moved Permanently`: The requested resource has been permanently moved to a new location. The response body usually contains the new location. NOTE: this normally means that the user is trying to access the server using HTTP and the server is configured for `FORCE_HTTPS==true`. The user should use HTTPS instead.

## 4xx (Client Errors)

- `400 Bad Request`: The request could not be understood by the server due to malformed syntax or invalid parameters. The response body often contains more specific error details.
- `401 Unauthorized`: Authentication is required and has failed or has not yet been provided. Ensure the auth token is valid and included in the request.
- `403 Forbidden`: Authentication was successful, but the authenticated user does not have permission to perform the requested action.
- `404 Not Found`: The requested resource could not be found.
- `405 Method Not Allowed`: The HTTP method used (e.g., GET, POST) is not supported for the requested resource.
- `408 Request Timeout`: The request timed out (often for async operations).
- `429 Too Many Requests`: The user has sent too many requests in a given amount of time (rate limiting).

## 5xx (Server Errors)

- `500 Internal Server Error`: An unexpected condition was encountered on the server.
- `503 Service Unavailable`: The server is currently unable to handle the request due to temporary overloading or maintenance.

When an error occurs, the response body will typically be a JSON object with a `detail` key:

```
{"detail": "Error message describing the issue"}
```

For validation errors (400 Bad Request), the response may include field-specific errors:

```
{
  "field_name": ["Error message for this field"],
  "another_field": ["Another error message"]
}
```

For server errors (500), the response may include an `error` key and optionally a `traceback`:

```
{
  "error": "An unexpected error occurred",
  "traceback": "...detailed traceback..."
}
```

# Pagination

LogZilla's API uses different pagination mechanisms depending on the endpoint:

## Standard List Pagination (Most Endpoints)

Used for endpoints like `/api/users`, `/api/dashboards`, and `/api/triggers`.

**Request:** Pass `page` and `page_size` as query parameters:

```
GET /api/users?page=2&page_size=50
```

**Response structure:**

```
{
  "objects": [ ... ],
  "item_count": 157,
  "page_count": 4,
  "page_number": 2
}
```

## Query Results Pagination

Used specifically for the `/api/query/{qid}` endpoint for event search results.

**Request:** Pass pagination parameters in the query body or as URL parameters:

```
{
  "page": 1,
  "page_size": 100,
  "offset": 0
}
```

**Response structure:** Pagination info is nested within the results:

```
{
  "query_id": "...",
  "status": "SUCCESS",
  "results": {
    "events": {
      "objects": [ ... ],
      "page_number": 1,
      "page_size": 100,
      "offset": 0,
```

```
      "item_count": 1234,
      "page_count": 13
    }
  }
}
```

# Common Data Structures and Formats

## Event Field Names

Events in LogZilla are characterized by several standard fields:

| Name | Description |
|---|---|
| `first_occurrence` | Timestamp of the first occurrence (epoch seconds with microseconds) |
| `last_occurrence` | Timestamp of the last occurrence (epoch seconds with microseconds) |
| `counter` | Number of occurrences in the deduplication window |
| `message` | The event message content |
| `host` | The originating host |
| `program` | The process or program name |
| `cisco_mnemonic` | Cisco mnemonic code (if applicable) |
| `severity` | Numeric severity (0-7, syslog standard) |
| `facility` | Numeric facility (0-23, syslog standard) |
| `status` | Status number (0=unknown, 1=actionable, 2=non-actionable) |
| `type` | Event type (e.g., `SYSLOG`, `INTERNAL`, `UNKNOWN`) |
| `User Tags` | User-defined fields |

# Best Practices

## Rate Limiting and Performance

- Use pagination for large result sets
- Cache query IDs for repeated access
- Implement exponential backoff for retries
- Use appropriate `page_size` values (typically 100-1000)

## Security

- Store API tokens securely (environment variables, secrets management)
- Use HTTPS for all API communications
- Implement proper error handling
- Regularly rotate API tokens

## Query Optimization

- Use specific filters to reduce result sets
- Leverage time ranges to limit data scope

# References

- Interactive API Documentation (`/api/docs` on the LogZilla server) - Current and detailed API specifications
- Authentication Guide (https://www.logzilla.ai/docs/logzilla-api/getting-started) - Token management and usage
- API Code Examples (https://www.logzilla.ai/docs/logzilla-api/sample-api-code) - Examples of performing API requests with error handling