LOGZILLA DOCUMENTATION

# Query Basics

Run structured searches against LogZilla from the command line using logzilla query with parameter files, authtoken setup, and query type selection

Command Line Tools · Generated May 3, 2026 · logzilla.ai/docs/command-line-tools/query-basics

© LogZilla Corporation · logzilla.ai

# LogZilla Query Command Basics

The `logzilla query` command provides programmatic access to LogZilla data through structured queries. This tool enables administrators to generate reports, extract specific data sets, and automate data analysis workflows.

# Getting Started

## Prerequisites

The query command requires authentication and proper permissions:

```
# Ensure you have root access or appropriate permissions
sudo su -

# Create API token for authentication
logzilla authtoken create --user admin
```

## Basic Command Structure

```
logzilla query --type <QueryType> --params <parameters-file> [options]
```

**Required Parameters:**

- `--type`: Specifies the query type (Search, TopN, EventRate, etc.)
- `--params`: Path to JSON file containing query parameters

**Common Options:**

- `--authtoken`: API authentication token
- `--user` / `--password`: Username/password authentication
- `--output-file`: Save results to file
- `--format`: Output format (json, xlsx)

## Authentication Setup

Method 1: API Token (Recommended)

```
# Create token and save to environment
TOKEN="$(logzilla authtoken create | tail -n1 | tr -d '[:space:]')"
export LOGZILLA_API_KEY="$TOKEN"

# Use in queries
logzilla query --type Search --params search.json --authtoken $LOGZILLA_API_KEY
```

Method 2: Configuration File

Create `~/.lz5query` configuration file:

```
[lz5query]
authtoken = your-api-token-here
base_url = http://localhost/api
```

Method 3: Username/Password

```
# Interactive password prompt
logzilla query --type Search --params search.json --user admin --password

# Direct password (less secure)
logzilla query --type Search --params search.json --user admin --password mypass
```

# Core Query Types

## Search Query

Retrieve specific events matching filter criteria. Most flexible query type for detailed event analysis.

**Parameters File (search-params.json):**

```
{
  "time_range": {
    "preset": "last_24_hours"
  },
  "filter": [
    {
      "field": "severity",
      "op": "le",
      "value": [3]
    }
  ],
  "limit": 100,
  "sort": ["-first_occurrence"]
}
```

**Execute Query:**

```
logzilla query --type Search --params search-params.json --authtoken $TOKEN
```

## TopN Query

Find the most frequent values for a specific field. Ideal for identifying top talkers, most active hosts, or common error patterns.

**Parameters File (topn-params.json):**

```
{
  "field": "host",
  "limit": 10,
  "time_range": {
    "preset": "last_7_days"
  },
  "filter": [
    {
      "field": "program",
      "op": "eq",
      "value": ["sshd"]
    }
  ]
}
```

**Execute Query:**

```
logzilla query --type TopN --params topn-params.json --authtoken $TOKEN
```

## EventRate Query

Calculate event rates over time periods. Useful for trend analysis and capacity planning.

**Parameters File (eventrate-params.json):**

```
{
  "time_range": {
    "preset": "last_24_hours",
    "step": 3600
  },
  "filter": [
    {
      "field": "severity",
      "op": "le",
```

```
      "value": [4]
    }
  ]
}
```

**Execute Query:**

```
logzilla query --type EventRate --params eventrate-params.json --authtoken $TOKEN
```

# Common Parameters

## Time Range Configuration

Time ranges define the period for data retrieval. Multiple formats supported:

Preset Time Ranges

```
{
  "time_range": {
    "preset": "today"
  }
}
```

**Available Presets:**

- `today` - Current day from midnight
- `yesterday` - Previous day
- `last_1_hours` - Last 1 hour
- `last_24_hours` - Last 24 hours
- `last_7_days` - Last 7 days
- `last_30_days` - Last 30 days
- `last_365_days` - Last 365 days

Note: Dynamic presets are supported with the pattern `last_N_unit`, where `unit` is one of `seconds`, `minutes`, `hours`, `days`, or `weeks`. Examples:

- `last_6_hours`
- `last_12_hours`
- `last_90_days`
- `last_2_weeks`

### Absolute Time Ranges

```
{
  "time_range": {
    "ts_from": "2024-01-01 00:00:00",
    "ts_to": "2024-01-31 23:59:59"
  }
}
```

### Unix Timestamps

```
{
  "time_range": {
    "ts_from": 1704067200,
    "ts_to": 1706745600
  }
}
```

### Time Steps for Aggregation

```
{
  "time_range": {
    "preset": "last_24_hours",
    "step": 3600
  }
}
```

**Common Step Values:**

- `60` - 1 minute intervals
- `900` - 15 minute intervals
- `3600` - 1 hour intervals
- `86400` - 1 day intervals

## Filter Configuration

Filters limit query results to specific criteria. Multiple filters are combined with AND logic.

### Basic Filter Structure

```
{
  "filter": [
    {
      "field": "field_name",
      "op": "operator",
      "value": ["value1", "value2"]
    }
```

```
  ]
}
```

Available Operators

| Operator | Description | Example |
|---|---|---|
| `eq` | Equals (default) | `"value": ["error"]` |
| `ne` | Not equals | `"value": ["debug"]` |
| `lt` | Less than | `"value": [5]` |
| `le` | Less than or equal | `"value": [3]` |
| `gt` | Greater than | `"value": [100]` |
| `ge` | Greater than or equal | `"value": [1000]` |
| `qp` | Query phrase (message search) | `"value": ["connection failed"]` |
| `=*` | Wildcard match | `"value": ["error*"]` |
| `!*` | Not wildcard match | `"value": ["debug*"]` |
| `=~` | Regex match | `"value": ["^ERROR.*"]` |
| `!~` | Not regex match | `"value": ["^DEBUG.*"]` |

Filter Examples

**Host Filter:**

```
{
  "field": "host",
  "op": "eq",
  "value": ["web-server-01", "web-server-02"]
}
```

**Severity Filter:**

```
{
  "field": "severity",
  "op": "le",
```

```
  "value": [3]
}
```

**Message Search:**

```
{
  "field": "message",
  "op": "qp",
  "value": ["authentication failed"]
}
```

**Wildcard Program Filter:**

```
{
  "field": "program",
  "op": "=*",
  "value": ["ssh*"]
}
```

## Archive Data Access

Include archived data in queries using the `with_archive` parameter:

```
{
  "time_range": {
    "preset": "last_365_days"
  },
  "with_archive": true,
  "filter": [
    {
      "field": "host",
      "op": "eq",
      "value": ["critical-server"]
    }
  ]
}
```

**Note**: Queries with `with_archive: true` may take longer to execute as they search both active and archived data.

## Result Limiting and Sorting

Limit Results

```
{
  "limit": 50,
```

```
  "page_size": 25
}
```

Sort Results

```
{
  "sort": ["-first_occurrence", "host"]
}
```

**Sort Fields:**

- `first_occurrence` / `-first_occurrence` (descending)
- `last_occurrence` / `-last_occurrence` (descending)
- `count` / `-count` (descending)
- `host`, `program`, `severity`

# Output Formats

## JSON Output (Default)

```
logzilla query --type Search --params search.json --output results.json
```

Results saved as structured JSON for programmatic processing.

## Excel Output

```
logzilla query --type TopN --params topn.json --output report.xlsx --format xlsx
```

Results formatted as Excel spreadsheet for business reporting.

## Console Output

```
logzilla query --type Search --params search.json
```

Results displayed directly in terminal for immediate review.

# Common Field Names

## Standard Event Fields

| Field | Description | Example Values |
|---|---|---|
| `host` | Source hostname | `web-server-01`, `192.168.1.100` |
| `program` | Program/service name | `sshd`, `kernel`, `nginx` |
| `message` | Log message text | `Connection established` |
| `severity` | Syslog severity (0-7) | `0` (Emergency) to `7` (Debug) |
| `facility` | Syslog facility (0-23) | `16` (Local0), `23` (Local7) |
| `first_occurrence` | First seen timestamp | `1704067200.123` |
| `last_occurrence` | Last seen timestamp | `1704067260.456` |
| `count` | Event occurrence count | `1`, `50`, `1000` |

## Enhanced Fields

| Field | Description | Example Values |
|---|---|---|
| `cisco_mnemonic` | Cisco device message ID | `ASA-6-302013` |
| `status` | Event processing status | `0` (Unknown), `1` (Known) |
| `user_tags` | Custom user tags | `{"environment": "production"}` |

# Error Handling

## Common Issues

**Authentication Errors:**

```
# Error: Invalid token
# Solution: Create new token
logzilla authtoken create --user admin
```

**Parameter Errors:**

```
# Error: Invalid JSON in parameters file
# Solution: Validate JSON syntax
cat params.json | jq .
```

**Time Range Errors:**

```
# Error: Invalid time range
# Solution: Check date format and range validity
```

## Debugging

```
# Enable debug output
logzilla query --type Search --params search.json --debug

# Validate parameters file
cat search.json | jq .

# Test with minimal parameters
echo '{"time_range": {"preset": "last_1_hours"}}' > test.json
logzilla query --type Search --params test.json
```

# Troubleshooting

Verify that the ~/.lz5query file is correct and contains the correct credentials.

```
[lz5query]
authtoken = your-api-token-here
base_url = http://localhost/api
```

# Best Practices

## Performance

- **Use specific time ranges** to limit data processing
- **Apply filters** to reduce result set size
- **Use appropriate limits** for large datasets
- **Consider archived data impact** on query performance

## Security

- **Use API tokens** instead of passwords
- **Protect configuration files** with proper permissions
- **Rotate tokens regularly** for security
- **Limit token permissions** to required operations

## Automation

- **Validate JSON parameters** before execution
- **Handle errors gracefully** in scripts
- **Log query execution** for audit trails
- **Use consistent naming** for parameter files

The query command provides the foundation for data analysis and reporting. Master these basics before moving to advanced query types and complex automation scenarios.

© LogZilla Corporation · logzilla.ai