LOGZILLA DOCUMENTATION

Receiving Events using HTTP

Send events to LogZilla over HTTPS at the /incoming endpoint using structured JSON or raw payloads, with interactive API docs and OpenAPI schema

Receiving Data in LogZilla · Generated April 27, 2026 · logzilla.ai/docs/receiving-data/http-event-receiver

HTTP Event Receiver

LogZilla has a "universal" facility to receive events via HTTP. This is called "universal" because it is not specific to any particular scenario — it is intended to be used with custom integrations.

LogZilla receives events over HTTP(S) on the UI/API port at path `/incoming`. The listener port is configured in Front settings (see Server Settings (https://www.logzilla.ai/docs/administration/server-settings)).

Interactive API docs are available at `/incoming/docs`.

Docs and schema

- Interactive docs: `/incoming/docs`
- OpenAPI schema: `/incoming/openapi.json`

Structured JSON Data Format

Recommended format of incoming data allows for best performance, as multiple events can be sent in single request. The events sent to LogZilla should to be formatted as JSON, with structure:

```
{
  "events": [
    // event1,
    // event2,
    // etc.
  ]
}
```

As the JSON array notation indicates, more than one event message can be sent per transmission, if desired. Then each event should have structure:

```
{
  "ts": 1704063600.1234,
  "host": "testhost.org",
  "program": "testprogram",
  "message": "this is the message",
  "user_tags": {
      "city": "Atlanta",
      "state": "Georgia"
  },
  "extra_fields": {
```

```
        "city": "Atlanta",
        "state": "Georgia"
    },
    "json": {
        "int_value": 1,
        "float_value": 1.1,
        "string_value": "foo",
        "object_value": {
          "foo" : "bar",
        },
        "array_value": ["bar", "baz"]
    }
}
```

Data Contents

The event fields that can be sent to LogZilla via HTTP are:

Field	Description
`ts`	epoch timestamp
`host`	the originating host of the log message
`program`	the program that generated the log message
`message`	log message
`priority`	number represents both the RFC-3164 Facility and Severity of the event in the message
`user_tags`	additional string fields that will be available as event attributes in both LogZilla rules and queries
`extra_fields`	additional string fields that will be available as event attributes in LogZilla rules
`json`	a special field that contains any json that will be available as event attribute in LogZilla rules

Unstructured JSON Data Format

If it's not possible to use the structured JSON format, then the raw JSON can be sent, by using `/incoming/raw` path. In this case, the JSON can contain any values, and it will be in the `extra_fields` of the message, and also in the serialized form in the `message` field. The `host` will be set to the IP address of the sender, and the `program` will be set to `http_receiver`.

This case is usually used with cooperation with some rules (usually from an app) that will extract interested fields from `extra_fields` and create appropriate event, depending on the actual content.

You can also use any subpath of `/incoming/raw`, like for example `/incoming/raw/app1`. The subpath will be available in the `extra_fields._url_path` field - in this example it will be `/app1`. This can be used in the rules to recognize events from different sources.

Authentication

When sending events (structured or unstructured), provide an access token using one of the supported methods:

- `Authorization: token YOUR_GENERATED_TOKEN`
- `X-LZ-Access-Key: YOUR_GENERATED_TOKEN`
- `AUTHTOKEN=YOUR_GENERATED_TOKEN` as a query parameter

See Obtaining an Auth Token (https://www.logzilla.ai/docs/logzilla-api/getting-started) for token management. If token changes are not picked up automatically, restart the module:

```
logzilla restart -c httpreceiver
```

Upon successful receipt of a JSON `events` data element, the HTTP receiver will respond with HTTP status code `202` and message:

```
{"detail": "ok"}
```

Examples

An example curl command using structured JSON:

```
curl \
  -H 'Content-Type: application/json' \
  -H 'Authorization: token YOUR_GENERATED_TOKEN' \
  -X POST -d  '{
      "events": [ {
        "message": "Test Message",
        "host": "curl.test",
        "program": "myapp",
        "extra_fields": { "city": "Atlanta", "state": "Georgia" },
        "json": { "int_value": 1, "string_value": "foo", "array_value": ["foo"] }
```

```
      } ] }' \
  'http://lzserver.company.com/incoming'
```

An example of using unstructured JSON:

```
curl \
  -H 'Content-Type: application/json' \
  -H 'Authorization: token YOUR_GENERATED_TOKEN' \
  -X POST -d  '{"foo": "bar"}' \
  'http://lzserver.company.com/incoming/raw/testapp'
```

In the latter case, the event will be created with `host` set to the IP address of the sender, `program` set to `http_receiver`, and `message` set to the `{"foo": "bar"}` string. Also the `extra_fields.foo` will contain `bar` and `extra_fields._url_path` will contain `/testapp`.

Verification

To verify end-to-end reception:

- Send a minimal test payload and include `-i` to show HTTP status:

```
curl -i \
  -H 'Content-Type: application/json' \
  -H 'Authorization: token YOUR_GENERATED_TOKEN' \
  -X POST -d '{"events":[{"message":"health check","host":"curl.test","program":"verify"}]}' \
  'http://lzserver.company.com/incoming'
```

Expected response contains `HTTP/1.1 202 Accepted` and body:

```
{"detail": "ok"}
```

Settings reference

For worker count, access logs, and invalid-request logging, see HTTP Receiver Settings (https://www.logzilla.ai/docs/administration/http-receiver-settings).

Compatibility endpoints

For backward compatibility, the receiver supports:

- `Content-Type: application/x-ndjson` to `/incoming` (rewritten to `/incoming/syslog-ndjson`).
- `/incoming/syslog-lines` for raw lines processing.

These endpoints are deprecated; use structured JSON whenever possible.

Health endpoints

- `/incoming/ping`
  - Lightweight check. Always returns `200` with body `{"status":"ok"}`.
  - Example:

```
curl -iS http://HOST:PORT/incoming/ping
```

- `/incoming/readiness`
  - Ready when the ingest pipeline is available. Returns `200` with `{"status":"ready"}`; otherwise returns `503` with `{"message":"Service temporarily unavailable"}`.
  - Example:

```
curl -iS http://HOST:PORT/incoming/readiness
```

- `/incoming/liveness`
  - Liveness probe. Always returns `200` with `{"status":"alive"}`.
  - Example:

```
curl -iS http://HOST:PORT/incoming/liveness
```