LOGZILLA DOCUMENTATION

Creating Custom Apps

Package LogZilla transformation rules, dashboards, triggers, and configuration schema as a custom app with a standard meta.yaml directory layout

Data Transforms · Generated April 29, 2026 · logzilla.ai/docs/data-transforms/creating-custom-apps

Creating custom apps

A LogZilla App packages transformation logic together with dashboards and triggers so parsing, enrichment, and visualization remain consistent. Apps are the preferred way to distribute vendor-specific parsing and to standardize field names and tags across environments.

Why create an app

- Ensure consistent parsing and enrichment across systems.
- Ship ready-to-use dashboards and widgets alongside rules.
- Include common triggers for alerting on important conditions.

What an app contains

- Transformation rules
  - Lua rules implement complex parsing and enrichment.
  - Rewrite rules supplement simple match-and-modify changes.
- Dashboards
  - Panels and widgets aligned to the normalized fields and tags.
- Triggers
  - Alert conditions that fire on important events or patterns.
- Configuration files
  - User-configurable settings that modify rule behavior without code changes.

App directory structure

A LogZilla App follows this standard directory structure:

```
my_custom_app/
├── meta.yaml                    # App metadata and description
├── config/
│   ├── config.yaml              # Default configuration values
│   └── config_schema.yaml       # Configuration validation schema
├── rules/
│   ├── 100-preprocessing.lua    # Lua transformation rules
│   └── 200-enrichment.lua       # (numbered for execution order)
├── dashboards/
│   └── main_dashboard.yaml      # Dashboard definitions
```

```
└── triggers/
    └── alert_rules.yaml        # Trigger/alert definitions
```

Required files

meta.yaml

Every app must include a `meta.yaml` file with basic metadata:

```yaml
---
name: My Custom App
description: Parses and enriches events from custom application
version: '1.0'
author: Your Organization
```

config/config_schema.yaml

Define the schema for user-configurable options:

```yaml
---
$id: my-custom-app-config
$schema: https://json-schema.org/draft/2020-12/schema
title: My Custom App Configuration
type: object
properties:
  enabled:
    type: boolean
    description: Enable event processing
    default: true
  severity_levels:
    type: object
    description: Map status codes to severity levels
    properties:
      error:
        type: string
        enum: ["High", "Medium", "Low"]
        default: "High"
```

Field and tag design

- Use clear, self-explanatory names (for example, `device_role`, `site`, `interface`).

Distribution and installation

Apps can be installed through the UI or command line:

Via UI (Settings → Applications)

- The App Store shows available and installed apps, details, and actions.
- Installing an app enables its rules, dashboards, and triggers.
- Uninstalling removes installed content; dependent apps may require prerequisites.

Via Command Line

```
# Create a new custom app (server will print the directory path as app_dir)
logzilla apps create my_custom_app

# Add your files (rules/, config/, dashboards/, triggers/) into the printed app_dir

# Validate the app without installing
logzilla apps test my_custom_app

# Install the app by code
logzilla apps install my_custom_app

# List installed apps
logzilla apps list

# Uninstall the app by code
logzilla apps uninstall my_custom_app
```

Configuration strategy

- Favor configuration-driven behavior for mappings and feature toggles.
- Validate configuration changes and support safe reload where available.
- Keep configuration small, documented, and scoped to the app's purpose.

Versioning and lifecycle

- Track changes with semantic versioning.
- Provide pre-install checks in documentation, including parent dependencies where applicable.
- Remove or archive deprecated dashboards and triggers as part of the upgrade steps.

Testing before rollout

- Validate and test rules using the CLI; see Testing and verification (https://www.logzilla.ai/docs/data-transforms/testing-and-verification).
- Use representative samples via device sources or the HTTP Event Receiver.
- Review parser metrics, rule errors, and field/tag values after enabling.
- Confirm dashboards render as intended and triggers fire on expected conditions.

Governance and security

- Treat the app as configuration-as-code; review and approve changes.
- Avoid emitting sensitive data in tags, messages, or dashboards.
- Align with organizational RBAC so dashboards and alerts reflect permitted data access.

Related reading

- Overview and decision guide (https://www.logzilla.ai/docs/data-transforms/transforming-event-streams)
- Lua rules (https://www.logzilla.ai/docs/data-transforms/lua-rules)
- Rewrite rules (https://www.logzilla.ai/docs/data-transforms/rewrite-rules)
- LogZilla Apps (https://www.logzilla.ai/docs/administration/logzilla-apps)
- Command Line Tools —- Data Commands (https://www.logzilla.ai/docs/command-line-tools/data-commands)