LOGZILLA DOCUMENTATION

# Trigger Scripts

Execute Python, Perl, shell, or compiled scripts from LogZilla triggers using EVENT_ and TRIGGER_ environment variables to access full event context

LogZilla Triggers · Generated June 12, 2026 · logzilla.ai/docs/creating-triggers/trigger-scripts

# Script Types

LogZilla can execute various types of scripts, including:

- Python
- Perl
- sh, bash, zsh, csh, etc.
- Compiled Executables

# Script Environment

All triggers passed to a script contain the matched message information as environment variables. To manipulate any of the data, simply reference the corresponding environment variable.

The following list of variables is automatically passed into each script:

```
# EVENT_ID                      =   <integer>
# EVENT_SEVERITY                =   <integer>
# EVENT_FACILITY                =   <integer>
# EVENT_MESSAGE                 =   <string>
# EVENT_HOST                    =   <string>
# EVENT_PROGRAM                 =   <string>
# EVENT_CISCO_MNEMONIC          =   <string>
# EVENT_USER_TAGS               =   <string>
# EVENT_STATUS                  =   <integer>
# EVENT_FIRST_OCCURRENCE        =   <float>
# EVENT_LAST_OCCURRENCE         =   <float>
# EVENT_COUNTER                 =   <integer>
# TRIGGER_ID                    =   <integer>
# TRIGGER_AUTHOR                =   <string>
# TRIGGER_AUTHOR_EMAIL          =   <string>
# TRIGGER_HITS_COUNT            =   <integer>
```

## Parsing User Tags

The `EVENT_USER_TAGS` environment variable contains all user tags as a single comma-separated string in the format: `key1=value1,key2=value2,key3=value3`

For example, if an event has user tags for `Username`, `SrcIP`, and `Auth Success`, the environment variable would contain:

```
EVENT_USER_TAGS="Username=admin,SrcIP=192.168.1.100,Auth Success=false"
```

### Bash Example: Parsing User Tags

```bash
#!/bin/bash

# Parse user tags from EVENT_USER_TAGS environment variable
parse_user_tags() {
    local tags="$1"
    local IFS=','
    for pair in $tags; do
        local key="${pair%%=*}"
        local value="${pair#*=}"
        case "$key" in
            Username) USERNAME="$value" ;;
            SrcIP) SRC_IP="$value" ;;
            "Auth Success") AUTH_SUCCESS="$value" ;;
            # Add more tags as needed
        esac
    done
}

# Use the function
parse_user_tags "$EVENT_USER_TAGS"

# Now you can use the extracted values
echo "Username: $USERNAME"
echo "Source IP: $SRC_IP"
echo "Auth Success: $AUTH_SUCCESS"
```

### Python Example: Parsing User Tags

```python
#!/usr/bin/env python3
import os

# Parse user tags from EVENT_USER_TAGS environment variable
def parse_user_tags(tags_string):
    """Parse comma-separated key=value pairs into a dictionary."""
    tags = {}
    if tags_string:
        for pair in tags_string.split(','):
            if '=' in pair:
                key, value = pair.split('=', 1)
                tags[key] = value
    return tags

# Get environment variable
event_user_tags = os.environ.get('EVENT_USER_TAGS', '')

# Parse into dictionary
user_tags = parse_user_tags(event_user_tags)

# Access individual tags
```

```
username = user_tags.get('Username', '')
src_ip = user_tags.get('SrcIP', '')
auth_success = user_tags.get('Auth Success', '')

print(f"Username: {username}")
print(f"Source IP: {src_ip}")
print(f"Auth Success: {auth_success}")
```

# Script Execution

Scripts may be executed directly or within dedicated Docker containers, depending on the script's requirements:

## Simple Scripts

For simple scripts that do not require anything beyond what is available in a standard Linux install, simply place the script in the `/etc/logzilla/scripts` directory and select it in the UI when creating a trigger.

Here's an example of a simple shell script that logs the environment variables to the `logzilla.log`:

Create a `test.sh` file in `/etc/logzilla/scripts/`:

```
cat << EOF > /etc/logzilla/scripts/test.sh
#!/bin/bash
# Print all environment variables matching '^EVENT_' to the log
echo "Test script env vars" >> /var/log/logzilla/logzilla.log
env | grep '^EVENT_' >> /var/log/logzilla/logzilla.log
EOF
```

Make sure the script is executable:

```
chmod 755 /etc/logzilla/scripts/test.sh
```

Reload script-server:

```
logzilla restart -c scriptserver
```

Once the script is in place and executable, users can select it from the LogZilla UI when creating a trigger.

## Custom Scripts

For scripts that require additional libraries or programs, such as Python packages, users may use their own Docker image containing all required modules.

## Working Example: Custom Docker Container

In this example, a container will be created that brings up an interface on a Cisco device after it is shut down, then send a notification to Slack. The script uses Python and Netmiko to SSH into the device and apply the necessary configuration changes.

Note: All of the files below are also available on the GitHub Repo (https://github.com/logzilla/extras/tree/master/howtos/trigger-cisco-config)

## Prepare custom image

Create a work directory used for Dockerfile and scripts

## Python Script

NOTE: The following sample code is user-contributed and should be reviewed prior to using it verbatim in production.

- Download or create `compliance.py` using the example from the GitHub repo (https://raw.githubusercontent.com/logzilla/extras/master/howtos/trigger-cisco-config/compliance.py)
- make the script executable

## Yaml and Slack Key

Create a `compliance.yaml` file and update the Slack webhook key. Edit the YAML configuration to fit the environment by updating the following variables:

```
# Cisco credentials
ciscoUsername: "cisco"
ciscoPassword: "cisco"

# Slack settings
# Replace the value below with your actual post URL
posturl: "https://hooks.slack.com/services/XXXXXXXXX/XXXXXXXXX/XXXXXXXXXXXXXXXXXXXXXXX"
default_channel: "#demo"
slack_user: "logzilla-bot"

# Logging and debug settings
log_file: "/var/log/logzilla/logzilla.log"

# Change to 0 in production:
debug_level: 2  # 0, 1, or 2

bring_interface_up: true

# Execution timeout for device connection and Slack:
timeout: 10
```

## Dockerfile

Create a new file named `Dockerfile` with the following content:

```
# Use a logzilla script-server base image
FROM logzilla/script-server:latest

# Copy the requirements.txt file to the container
COPY requirements.txt /tmp/requirements.txt

# Install Python dependencies
RUN pip install -r /tmp/requirements.txt \
    --no-cache-dir --break-system-packages --root-user-action=ignore

# Copy script content to the container
RUN mkdir -p /scripts
COPY compliance.py /scripts
COPY compliance.yaml /scripts
RUN chmod +x /scripts/compliance.py
```

## Requirements.txt

Create a `requirements.txt` file with the following content:

```
netmiko
paramiko
pyyaml
requests
```

## Docker compose file

Create a `compose.yaml` file with the following content:

```
services:
  api:
    build:
      context: .
    container_name: compliance-script-server
    environment:
      SCRIPTS_ENABLED: "1"
      SCRIPTS_DIR: /scripts
      SCRIPTS_LOGS_DIR: /var/log/script-logs
    volumes:
        - logs:/var/log/script-logs/
    networks:
      - lz_network
```

```
volumes:
  logs:

networks:
  lz_network:
    name: lz_main
    external: true
```

## Directory Contents

The work directory should contain the following files:

- Dockerfile
- compliance.py
- compliance.yaml
- compose.yaml
- requirements.txt

## Run custom script container using docker compose

```
docker compose up --build -d
```

## Check containers is running

```
# docker ps -a -f name=compliance-script-server
CONTAINER ID   IMAGE                              COMMAND                  CREATED         STATUS
PORTS     NAMES
e55547cfb505   custom-trigger-cisco-compliance   "fastapi run /usr/li..."   7 seconds ago   Up 7
seconds             compliance-script-server
```

## Register custom script container

Create or edit `/etc/logzilla/settings/script_server.yaml`:

```
---
SERVERS:
  - name: custom
    url: http://compliance-script-server:8000/scripts
```

Reload LogZilla settings:

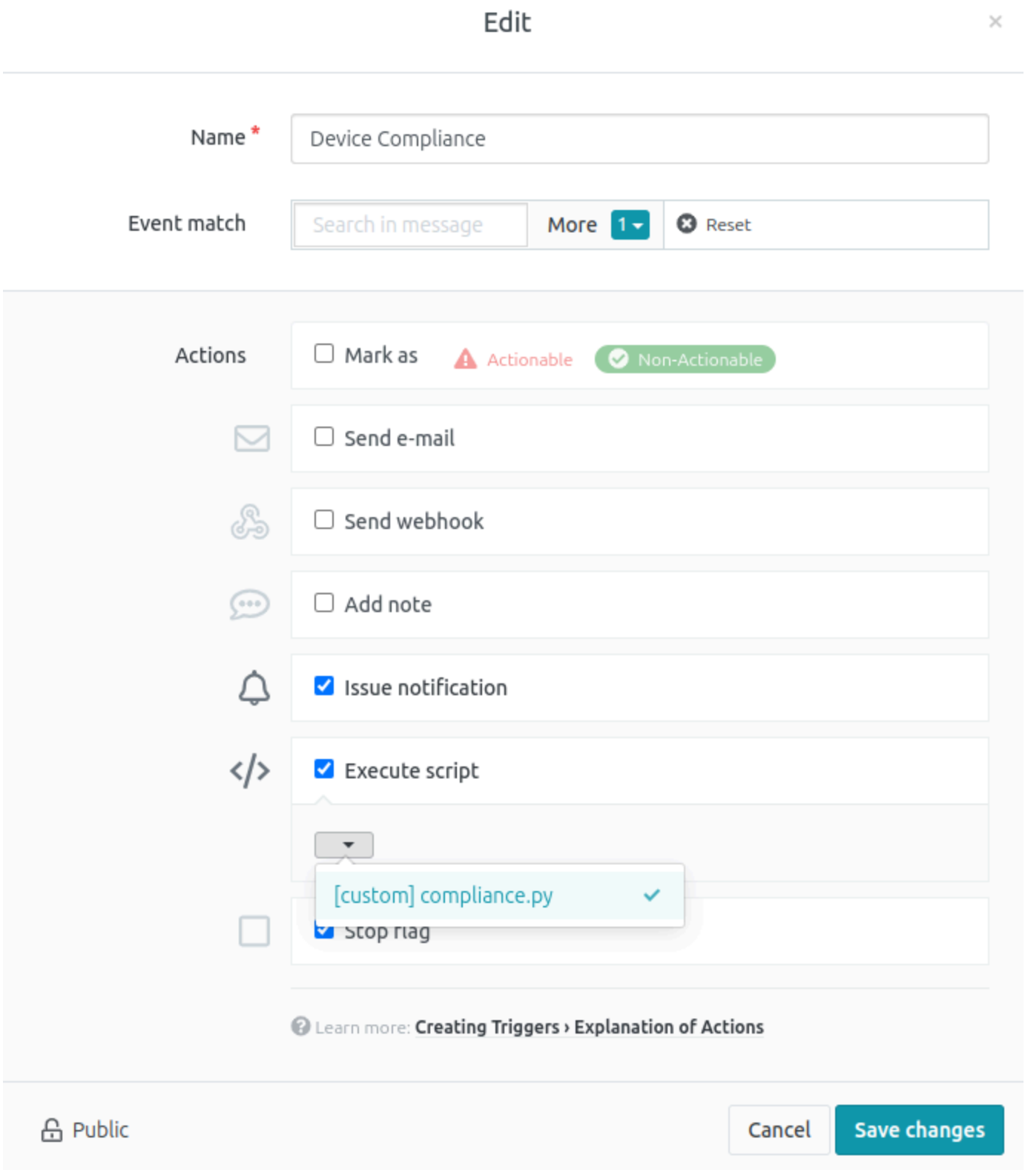
```
logzilla settings reload script_server
```

## LogZilla UI

Log into the LogZilla Web Interface and:

- Create a new trigger from the trigger menu.
- Select the `execute script` option.
- From the dropdown menu, select `[custom] compliance.py`.

Any patterns matching this trigger will now execute the script.


Edit
Name *
Device Compliance
Event match
Search in message
More 1
Reset
Actions
Mark as
Actionable
Non-Actionable
Send e-mail
Send webhook
Add note
Issue notification
Execute script
[custom] compliance.py
Stop flag
Learn more: Creating Triggers › Explanation of Actions
Public
Cancel
Save changes