LOGZILLA DOCUMENTATION

Automations

Automate LogZilla responses with SOAR-style trigger actions that run scripts, post webhooks, send email, and remediate events like Cisco interface outages

Alerts · Generated June 12, 2026 · logzilla.ai/docs/alerts/automations

LogZilla's automation capabilities provide Security Orchestration, Automation, and Response (SOAR) functionality that enables organizations to respond to events automatically, reducing manual intervention and improving response times. Users can create automated workflows that execute scripts, send webhooks, generate notifications, and orchestrate responses across multiple systems.

How Automation Works

LogZilla automation operates through triggers that execute actions when specific events occur. When a trigger matches an event, it can:

- Execute custom scripts with full event context
- Send HTTP webhooks to external systems
- Generate email notifications with event details
- Create internal notifications and alerts
- Mark events for classification and filtering

All automation actions receive complete event information through environment variables, enabling context-aware responses.

Practical Automation Examples

The following examples demonstrate real-world automation scenarios users can implement:

Network Interface Recovery

Scenario: A Cisco interface goes down unexpectedly.

Automation: A trigger detects the `%LINK-3-UPDOWN` event and executes a Python script that:

Uses python to SSH into the device
Runs `no shutdown` on the affected interface
Sends a Slack notification confirming the remediation
Implementation: The script receives the device hostname via the `EVENT_HOST` environment variable and interface details parsed from `EVENT_MESSAGE`.

Duplex Mismatch Correction

Scenario: A CDP duplex mismatch is detected on a network link.

Automation: When LogZilla receives a `%CDP-4-DUPLEX_MISMATCH` event, a trigger executes a script that:

Connects to both devices involved in the mismatch
Configures both interfaces to use the same duplex setting
Logs the corrective action taken

Service Restart with Threshold

Scenario: A critical service fails and needs automatic recovery with escalation limits.

Automation: Using event correlation rules, the system:

Detects service failure events
Attempts to restart the service automatically
Tracks restart attempts (maximum 3 times in 5 minutes)
Escalates to human operators if automatic recovery fails

Flapping Interface Detection

Scenario: A network interface is unstable, repeatedly going up and down.

Automation: Event correlation identifies the pattern:

Correlates interface down/up event pairs within a time window
Counts "short outage" occurrences
Generates an alert when flapping exceeds threshold (e.g., 10 times in 6 hours)
Notifies network team of unstable interface requiring investigation

Security Incident Response

Scenario: Multiple failed login attempts detected from a single source.

Automation: A trigger identifies the pattern and:

Automatically disables the affected user account
Creates a security incident ticket in ServiceNow
Sends alert to security team with user and source IP details
Blocks the source IP at the firewall if attempts continue

Malware Communication Detection

Scenario: A host makes numerous outbound connections to suspicious destinations.

Automation: When firewall logs show high-volume connection attempts:

Cross-reference destination IPs with threat intelligence feeds

Automatically block confirmed malicious IPs at the firewall
Quarantine the source host via NAC integration
Create high-priority security incident for investigation

ServiceNow Integration

Scenario: Automatically create and update incident tickets for critical events.

Automation: A trigger sends HTTP POST requests to ServiceNow's REST API:

Creates incident tickets with event details
Uses correlation IDs to prevent duplicate tickets
Updates existing tickets when related events occur
Includes event context, severity, and affected systems
Implementation: The webhook payload includes LogZilla variables like `{{event:host}}`, `{{event:message}}`, and `{{event:severity}}` to populate ServiceNow fields automatically.

Interactive Slack Automation

Scenario: Network interface down with human-in-the-loop remediation.

Automation: When an interface goes down:

Sends Slack message with interface details and "Remediate" button
Operator clicks button to authorize automatic remediation
LogZilla receives postback and executes interface recovery script
Sends confirmation message with remediation results

Implementation Methods

Script Execution

Scripts receive complete event context through environment variables:

```
EVENT_HOST=router01.company.com
EVENT_MESSAGE=%LINK-3-UPDOWN: Interface GigabitEthernet0/1, changed state to down
EVENT_SEVERITY=3
EVENT_FACILITY=23
EVENT_PROGRAM=LINK
```

Scripts can use this information to make intelligent, context-aware decisions about remediation actions.

Event Correlation

Complex automation scenarios use the Simple Event Correlator (SEC) to:

- **PairWithWindow**: Match related events within time windows
- **SingleWithThreshold**: Count occurrences and trigger on thresholds
- **EventGroup**: Correlate multiple event types for complex patterns

Webhook Integration

Send structured data to external APIs using HTTP POST requests with JSON payloads containing event variables and custom fields.

LogZilla SOAR Advantages

LogZilla has provided SOAR capabilities for years before the term "Security Orchestration, Automation, and Response" became widely adopted. This established foundation offers several advantages:

- **Proven Reliability**: Battle-tested automation engine with years of production deployment
- **High Performance**: Purpose-built for high-volume event processing and real-time response
- **Flexible Integration**: Extensive API and webhook support for security tool orchestration
- **Cost Effective**: SOAR functionality without the complexity and licensing costs of dedicated SOAR platforms
- **Unified Platform**: Combines log management, SIEM capabilities, and SOAR automation in a single solution

Getting Started

1. Identify Automation Opportunities

Look for repetitive manual tasks triggered by specific events:

- Service restarts after failures
- Network device configuration issues
- Security incidents requiring immediate response
- Ticket creation for critical alerts

2. Start Simple

Begin with basic automations before building complex workflows:

- Email notifications for critical events
- Slack alerts with event details
- Simple script execution for common fixes

3. Build Context-Aware Scripts

Use environment variables to create intelligent automations:

- `EVENT_HOST` for device-specific actions
- `EVENT_MESSAGE` for parsing specific details
- `EVENT_SEVERITY` for response prioritization
- `EVENT_USER_TAGS` for enriched context

4. Implement Safeguards

Include error handling and limits in automation scripts:

- Maximum retry attempts with backoff
- Validation before making changes
- Logging of all automated actions
- Escalation paths when automation fails

Advanced Scenarios

Domain Trust Enumeration Detection

Detect reconnaissance activity by correlating events from multiple domain controllers when a single source attempts to enumerate domain trusts across several systems within a short timeframe.

Windows Event Log Service Monitoring

Monitor for Windows Event Log service failures and automatically restart the service, escalating to administrators if the service fails to stay running.

Technical Implementation

For detailed technical information on implementing automation scripts, including environment variables, script requirements, and Docker container setup, see the Trigger Scripts (https://www.logzilla.ai/docs/creating-triggers/trigger-scripts) documentation.

For complex event correlation scenarios, see the Event Correlation (https://www.logzilla.ai/docs/event-correlation/event-correlation-overview) documentation.