Notification handling

- How the system handles and works around with notifications.

- Notifications & Triggers

Notifications & Triggers

CFM Notifications System

Overview

The **CFM Notifications System** monitors the availability of network agents and provides customizable alerting when agents go offline or recover. It also integrates with **user-triggered unblock request monitoring**, helping administrators stay informed about potential abuse or false positives in blocking behavior.

The system is composed of four coordinated components:

1. **Agent Uptime Monitoring Controller**
2. **Notifier Configuration Resource**
3. **Trigger Mechanism Resource**
4. **Unblock Request Monitoring**

These components work together to detect agent status changes, define notification methods, and execute appropriate alerting workflows.

---

1. Agent Uptime Monitoring

Component: `AgentNotificationController`

This controller is responsible for continuously checking the status of registered agents. It does so by comparing each agent's last heartbeat timestamp (`last_seen_at`) against the current time. If an agent hasn't communicated within a predefined time window (e.g., 60 seconds), it is considered **offline**.

Key Behaviors

- **Recovery Detection**: If an agent previously marked as offline starts reporting again, the system logs a restoration event.

- **Outage Detection**: Agents not seen in the defined window are marked as down.
- **Notification Throttling**: Notifications are suppressed if a recent alert was already sent (e.g., within the past hour), avoiding redundant notifications.

---

2. □□ Notifier Configuration

Component: `NotifierResource`

This resource allows administrators to configure various **notification channels**, which define **how alerts are sent**.

Supported Notifier Types (examples):

- Email
- Slack
- Webhook
- Custom API callouts

Each notifier contains:

- A unique name.
- A target configuration (e.g., email address or webhook URL).
- A type identifier (used to trigger the appropriate method).
- Optionally, a list of tags or filters to determine relevance to specific agents.

---

3. □□ Trigger Mechanism

Component: `TriggerResource`

This module acts as the **brain of the notification system**, determining **when and which notifications** should be triggered based on agent status changes.

Core Features

- **Linkage** between triggers and notifiers (one-to-many).
- **Conditions**: Triggers can be configured to react only to specific types of status changes (e.g., only on downtime, or both up/down events).

- **Tag Matching**: Allows targeting subsets of agents based on metadata.
- **Last Notified Tracking**: Stores timestamps of previous alerts per trigger-agent pair to control re-alerting frequency.

---

4. □□ Unblock Request Monitoring

Integration Point: `UnblockController` (external module)

This integration captures and processes **unblock requests submitted by end users** who are temporarily blocked by the system (e.g., via CSF or custom firewall logic).

Features

- **Alerting on Unblock Attempts**: Notifies admins when a user requests to be unblocked.
- **Request Outcome Visibility**: Indicates if the user:
  - Was indeed blocked and unblocked.
  - Was never blocked.
  - Has requested unblocking too frequently (possible abuse).
- **Escalation Signals**: Excessive or suspicious unblock activity can flag issues for deeper inspection.

This feature provides security teams with real-time context on potentially malicious or misbehaving clients.

□□ Security Considerations

To protect system integrity and minimize attack surface:

- Internal logic is abstracted and modularized to reduce exposure.
- Logging is selectively enabled and avoids sensitive data leakage.
- Time-based checks and cooldown intervals prevent alert spam or abuse.
- All external interactions (e.g., sending to Slack or webhooks) should be validated and rate-limited.
- All external outbound notifications should be validated and rate-limited.
- Unblock request alerts help detect misuse or false positive blocks.

⚙ Workflow Summary

1. The **Agent Uptime Controller** routinely checks agent heartbeat timestamps.
2. If a status change is detected (up or down), it invokes the **Trigger Mechanism**.
3. The **Trigger Mechanism** determines if and which notifications should be sent, based on tag matching and cooldown logic.
4. Matched **Notifiers** are executed asynchronously or in queue, depending on system setup.
5. If a user **requests unblocking**, an internal event is logged and optionally notifies admins based on thresholds or flags.