Auto-enrolment api documentation

Overview

This document provides comprehensive API documentation for PatchMon's auto-enrollment system, covering token management, host enrollment, and agent installation endpoints. These APIs enable automated device onboarding using tools like Ansible, Terraform, or custom scripts.

Table of Contents

API Architecture

Base URL Structure

```
https://your-patchmon-server.com/api/v1/
```

The API version is configurable via the `API_VERSION` environment variable (defaults to `v1`).

Endpoint Categories

Category	Path Prefix	Authentication	Purpose
Admin	`/auto-enrollment/tokens/*`	JWT (Bearer token)	Token management (CRUD)
Enrollment	`/auto-enrollment/*`	Token key + secret (headers)	Host enrollment & script download
Host	`/hosts/*`	API ID + key (headers)	Agent installation & data reporting

Two-Tier Security Model

Tier 1: Auto-Enrollment Token

- **Purpose**: Create new host entries via enrollment
- **Scope**: Limited to enrollment operations only
- **Authentication**: `X-Auto-Enrollment-Key` + `X-Auto-Enrollment-Secret` headers
- **Rate Limited**: Yes (configurable hosts per day per token)
- **Storage**: Secret is hashed (bcrypt) in the database

Tier 2: Host API Credentials

- **Purpose**: Agent communication (data reporting, updates, commands)
- **Scope**: Per-host unique credentials
- **Authentication**: `X-API-ID` + `X-API-KEY` headers
- **Rate Limited**: No (per-host)
- **Storage**: API key is hashed (bcrypt) in the database

Why two tiers?

- Compromised enrollment token ≠ compromised hosts
- Compromised host credential ≠ compromised enrollment
- Revoking an enrollment token stops new enrollments without affecting existing hosts

Authentication

Admin Endpoints (JWT)

All admin endpoints require a valid JWT Bearer token from an authenticated user with "Manage Settings" permission:

```
curl -H "Authorization: Bearer <jwt_token>" \
     -H "Content-Type: application/json" \
     https://your-patchmon-server.com/api/v1/auto-enrollment/tokens
```

Enrollment Endpoints (Token Key + Secret)

Enrollment endpoints authenticate via custom headers:

```
curl -H "X-Auto-Enrollment-Key: patchmon_ae_abc123..." \
     -H "X-Auto-Enrollment-Secret: def456ghi789..." \
     -H "Content-Type: application/json" \
     https://your-patchmon-server.com/api/v1/auto-enrollment/enroll
```

Host Endpoints (API ID + Key)

Host endpoints authenticate via API credential headers:

```
curl -H "X-API-ID: patchmon_abc123" \
     -H "X-API-KEY: def456ghi789" \
     https://your-patchmon-server.com/api/v1/hosts/install
```

Admin Endpoints

All admin endpoints require JWT authentication and "Manage Settings" permission.

Create Auto-Enrollment Token

Endpoint: `POST /api/v1/auto-enrollment/tokens`

Request Body:

Field	Type	Required	Default	Description
`token_name`	string	Yes	—	Descriptive name (max 255 chars)
`max_hosts_per_day`	integer	No	`100`	Rate limit (1–1000)
`default_host_group_id`	string	No	`null`	UUID of host group to auto-assign
`allowed_ip_ranges`	string[]	No	`[]`	IP whitelist (exact IPs or CIDR notation)
`expires_at`	string	No	`null`	ISO 8601 expiration date

Field	Type	Required	Default	Description
`metadata`	object	No	`{}`	Custom metadata (e.g. `integration_type`, `environment`)
`scopes`	object	No	`null`	Permission scopes (only for API integration type tokens)

Example Request:

```
{
  "token_name": "Proxmox Production",
  "max_hosts_per_day": 100,
  "default_host_group_id": "uuid-of-host-group",
  "allowed_ip_ranges": ["192.168.1.10", "10.0.0.0/24"],
  "expires_at": "2026-12-31T23:59:59Z",
  "metadata": {
    "integration_type": "proxmox-lxc",
    "environment": "production"
  }
}
```

Response: `201 Created`

```
{
  "message": "Auto-enrollment token created successfully",
  "token": {
    "id": "uuid",
    "token_name": "Proxmox Production",
    "token_key": "patchmon_ae_abc123...",
    "token_secret": "def456ghi789...",
    "max_hosts_per_day": 100,
    "default_host_group": {
      "id": "uuid",
      "name": "Proxmox LXC",
      "color": "#3B82F6"
    },
    "created_by": {
      "id": "uuid",
      "username": "admin",
```

```
      "first_name": "John",
      "last_name": "Doe"
    },
    "expires_at": "2026-12-31T23:59:59Z",
    "scopes": null
  },
  "warning": "⚠ Save the token_secret now - it cannot be retrieved later!"
}
```

> **Important:** The `token_secret` is only returned in this response. It is hashed before storage and cannot be retrieved again.

List Auto-Enrollment Tokens

Endpoint: `GET /api/v1/auto-enrollment/tokens`

Response: `200 OK`

```
[
  {
    "id": "uuid",
    "token_name": "Proxmox Production",
    "token_key": "patchmon_ae_abc123...",
    "is_active": true,
    "allowed_ip_ranges": ["192.168.1.10"],
    "max_hosts_per_day": 100,
    "hosts_created_today": 15,
    "last_used_at": "2025-10-11T14:30:00Z",
    "expires_at": "2026-12-31T23:59:59Z",
    "created_at": "2025-10-01T10:00:00Z",
    "default_host_group_id": "uuid",
    "metadata": { "integration_type": "proxmox-lxc" },
    "scopes": null,
    "host_groups": {
      "id": "uuid",
      "name": "Proxmox LXC",
      "color": "#3B82F6"
    },
    "users": {
```

```
      "id": "uuid",
      "username": "admin",
      "first_name": "John",
      "last_name": "Doe"
    }
  }
]
```

Tokens are returned in descending order by creation date. The `token_secret` is never included in list responses.

Get Token Details

Endpoint: `GET /api/v1/auto-enrollment/tokens/{tokenId}`

Response: `200 OK` — Same structure as a single token in the list response (without `token_secret`).

Error: `404 Not Found` if `tokenId` does not exist.

Update Token

Endpoint: `PATCH /api/v1/auto-enrollment/tokens/{tokenId}`

All fields are optional — only include fields you want to change.

Request Body:

Field	Type	Description
`token_name`	string	Updated name (1–255 chars)
`is_active`	boolean	Enable or disable the token
`max_hosts_per_day`	integer	Updated rate limit (1–1000)
`allowed_ip_ranges`	string[]	Updated IP whitelist
`default_host_group_id`	string	Updated host group (set to empty string to clear)
`expires_at`	string	Updated expiration date (ISO 8601)
`scopes`	object	Updated scopes (API integration type tokens only)

Example Request:

```
{
  "is_active": false,
  "max_hosts_per_day": 200,
  "allowed_ip_ranges": ["192.168.1.0/24"]
}
```

Response: `200 OK`

```
{
  "message": "Token updated successfully",
  "token": {
    "id": "uuid",
    "token_name": "Proxmox Production",
    "token_key": "patchmon_ae_abc123...",
    "is_active": false,
    "max_hosts_per_day": 200,
    "allowed_ip_ranges": ["192.168.1.0/24"],
    "host_groups": { "id": "uuid", "name": "Proxmox LXC", "color": "#3B82F6" },
    "users": { "id": "uuid", "username": "admin", "first_name": "John", "last_name": "Doe" }
  }
}
```

Errors:

- `404 Not Found` — Token does not exist
- `400 Bad Request` — Host group not found, or scopes update attempted on a non-API token

Delete Token

Endpoint: `DELETE /api/v1/auto-enrollment/tokens/{tokenId}`

Response: `200 OK`

```
{
  "message": "Auto-enrollment token deleted successfully",
  "deleted_token": {
    "id": "uuid",
    "token_name": "Proxmox Production"
  }
}
```

Error: `404 Not Found` if `tokenId` does not exist.

Enrollment Endpoints

Download Enrollment Script

Endpoint: `GET /api/v1/auto-enrollment/script`

This endpoint validates the token credentials, then serves a bash script with the PatchMon server URL, token credentials, and configuration injected automatically.

Query Parameters:

Parameter	Required	Description
`type`	Yes	Script type: `proxmox-lxc` or `direct-host`
`token_key`	Yes	Auto-enrollment token key
`token_secret`	Yes	Auto-enrollment token secret
`force`	No	Set to `true` to enable force install mode (for broken packages)

Example:

```
curl "https://patchmon.example.com/api/v1/auto-enrollment/script?type=proxmox-lxc&token_key=KEY&token_secret=SECRET"
```

Response: `200 OK` — Plain text bash script with credentials injected.

Errors:

- `400 Bad Request` — Missing or invalid `type` parameter
- `401 Unauthorized` — Missing credentials, invalid/inactive token, invalid secret, or expired token
- `404 Not Found` — Script file not found on server

Enroll Single Host

Endpoint: `POST /api/v1/auto-enrollment/enroll`

Headers:

```
X-Auto-Enrollment-Key: patchmon_ae_abc123...
X-Auto-Enrollment-Secret: def456ghi789...
Content-Type: application/json
```

Request Body:

Field	Type	Required	Description
`friendly_name`	string	Yes	Display name for the host (max 255 chars)
`machine_id`	string	No	Unique machine identifier (max 255 chars)
`metadata`	object	No	Additional metadata (vmid, proxmox_node, ip_address, os_info, etc.)

Example Request:

```
{
  "friendly_name": "webserver",
  "machine_id": "proxmox-lxc-100-abc123",
  "metadata": {
    "vmid": "100",
    "proxmox_node": "proxmox01",
    "ip_address": "10.0.0.10",
    "os_info": "Ubuntu 22.04 LTS"
  }
}
```

Response: `201 Created`

```
{
  "message": "Host enrolled successfully",
  "host": {
    "id": "uuid",
    "friendly_name": "webserver",
    "api_id": "patchmon_abc123def456",
    "api_key": "raw-api-key-value",
    "host_group": {
      "id": "uuid",
      "name": "Proxmox LXC",
      "color": "#3B82F6"
```

```
    },
    "status": "pending"
  }
}
```

> **Note:** The `api_key` is only returned in this response (plain text). It is hashed before storage. The `host_group` is `null` if no default host group is configured on the token.

Error Responses:

Status	Error	Cause
`400`	Validation errors	Missing or invalid `friendly_name`
`401`	`Auto-enrollment credentials required`	Missing `X-Auto-Enrollment-Key` or `X-Auto-Enrollment-Secret` headers
`401`	`Invalid or inactive token`	Token key not found or token is disabled
`401`	`Invalid token secret`	Secret does not match
`401`	`Token expired`	Token has passed its expiration date
`403`	`IP address not authorized for this token`	Client IP not in `allowed_ip_ranges`
`429`	`Rate limit exceeded`	Token's `max_hosts_per_day` limit reached

> **Duplicate handling:** The API does not perform server-side duplicate host checks. Duplicate prevention is handled client-side by the enrollment script, which checks for an existing agent configuration (`/etc/patchmon/config.yml`) inside each container before calling the API.

Bulk Enroll Hosts

Endpoint: `POST /api/v1/auto-enrollment/enroll/bulk`

Headers:

```
X-Auto-Enrollment-Key: patchmon_ae_abc123...
X-Auto-Enrollment-Secret: def456ghi789...
Content-Type: application/json
```

Request Body:

```
{
  "hosts": [
    {
      "friendly_name": "webserver",
      "machine_id": "proxmox-lxc-100-abc123"
    },
    {
      "friendly_name": "database",
      "machine_id": "proxmox-lxc-101-def456"
    }
  ]
}
```

Limits:

- Minimum: 1 host per request
- Maximum: 50 hosts per request
- Each host must have a `friendly_name` (required); `machine_id` is optional

Response: `201 Created`

```
{
  "message": "Bulk enrollment completed: 2 succeeded, 0 failed, 0 skipped",
  "results": {
    "success": [
      {
        "id": "uuid",
        "friendly_name": "webserver",
        "api_id": "patchmon_abc123",
        "api_key": "def456"
      },
      {
        "id": "uuid",
        "friendly_name": "database",
        "api_id": "patchmon_ghi789",
        "api_key": "jkl012"
      }
    ],
    "failed": [],
```

```
    "skipped": []
  }
}
```

Rate Limit Error (`429`):

```
{
  "error": "Rate limit exceeded",
  "message": "Only 5 hosts remaining in daily quota"
}
```

The bulk endpoint checks the remaining daily quota before processing. If the number of hosts in the request exceeds the remaining quota, the entire request is rejected.

Host Management Endpoints

These endpoints are used by the PatchMon agent (not the enrollment script). They authenticate using the per-host `X-API-ID` and `X-API-KEY` credentials returned during enrollment.

Download Agent Installation Script

Endpoint: `GET /api/v1/hosts/install`

Serves a shell script that bootstraps the PatchMon agent on a host. The script uses a secure bootstrap token mechanism — actual API credentials are not embedded directly in the script.

Headers:

```
X-API-ID: patchmon_abc123
X-API-KEY: def456ghi789
```

Query Parameters:

Parameter	Required	Description
`force`	No	Set to `true` to enable force install mode
`arch`	No	Architecture override (e.g. `amd64`, `arm64`); auto-detected if omitted

Response: `200 OK` — Plain text shell script with bootstrap token injected.

Download Agent Binary/Script

Endpoint: `GET /api/v1/hosts/agent/download`

Downloads the PatchMon agent binary (Go binary for modern agents) or migration script (for legacy bash agents).

Headers:

```
X-API-ID: patchmon_abc123
X-API-KEY: def456ghi789
```

Query Parameters:

Parameter	Required	Description
`arch`	No	Architecture (e.g. `amd64`, `arm64`)
`force`	No	Set to `binary` to force binary download

Response: `200 OK` — Binary file or shell script.

Host Data Update

Endpoint: `POST /api/v1/hosts/update`

Used by the agent to report package data, system information, and hardware details.

Headers:

```
X-API-ID: patchmon_abc123
X-API-KEY: def456ghi789
Content-Type: application/json
```

Request Body Fields:

Field	Type	Required	Description
`packages`	array	Yes	Array of package objects (max 10,000)
`packages[].name`	string	Yes	Package name
`packages[].currentVersion`	string	Yes	Currently installed version
`packages[].availableVersion`	string	No	Available update version

Field	Type	Required	Description
`packages[].needsUpdate`	boolean	Yes	Whether an update is available
`packages[].isSecurityUpdate`	boolean	No	Whether the update is security-related
`agentVersion`	string	No	Reporting agent version
`osType`	string	No	Operating system type
`osVersion`	string	No	Operating system version
`hostname`	string	No	System hostname
`ip`	string	No	System IP address
`architecture`	string	No	CPU architecture
`cpuModel`	string	No	CPU model name
`cpuCores`	integer	No	Number of CPU cores
`ramInstalled`	float	No	Installed RAM in GB
`swapSize`	float	No	Swap size in GB
`diskDetails`	array	No	Array of disk objects
`gatewayIp`	string	No	Default gateway IP
`dnsServers`	array	No	Array of DNS server IPs
`networkInterfaces`	array	No	Array of network interface objects
`kernelVersion`	string	No	Running kernel version
`installedKernelVersion`	string	No	Installed (on-disk) kernel version
`selinuxStatus`	string	No	SELinux status (`enabled`, `disabled`, or `permissive`)
`systemUptime`	string	No	System uptime
`loadAverage`	array	No	Load average values
`machineId`	string	No	Machine ID
`needsReboot`	boolean	No	Whether a reboot is required
`rebootReason`	string	No	Reason a reboot is required
`repositories`	array	No	Configured package repositories
`executionTime`	string	No	Time taken to gather data

Example Request:

```
{
  "packages": [
    {
      "name": "nginx",
      "currentVersion": "1.18.0",
      "availableVersion": "1.20.0",
      "needsUpdate": true,
      "isSecurityUpdate": false
    }
  ],
  "agentVersion": "1.2.3",
  "cpuModel": "Intel Xeon E5-2680 v4",
  "cpuCores": 8,
  "ramInstalled": 16.0,
  "swapSize": 2.0,
  "diskDetails": [
    {
      "device": "/dev/sda1",
      "mountPoint": "/",
      "size": "50GB",
      "used": "25GB",
      "available": "25GB"
    }
  ],
  "gatewayIp": "192.168.1.1",
  "dnsServers": ["8.8.8.8", "8.8.4.4"],
  "networkInterfaces": [
    {
      "name": "eth0",
      "ip": "192.168.1.10",
      "mac": "00:11:22:33:44:55"
    }
  ],
  "kernelVersion": "5.4.0-74-generic",
  "selinuxStatus": "disabled"
}
```


Response: `200 OK`

```
{
  "message": "Host updated successfully",
  "packagesProcessed": 1,
  "updatesAvailable": 1,
  "securityUpdates": 0
}
```

Ansible Integration Examples

Basic Playbook for Proxmox Enrollment

```
---
- name: Enroll Proxmox LXC containers in PatchMon
  hosts: proxmox_hosts
  become: yes
  vars:
    patchmon_url: "https://patchmon.example.com"
    token_key: "{{ vault_patchmon_token_key }}"
    token_secret: "{{ vault_patchmon_token_secret }}"
    host_prefix: "prod-"

  tasks:
    - name: Install dependencies
      apt:
        name:
          - curl
          - jq
        state: present

    - name: Download enrollment script
      get_url:
        url: "{{ patchmon_url }}/api/v1/auto-enrollment/script?type=proxmox-lxc&token_key={{
token_key }}&token_secret={{ token_secret }}"
        dest: /root/proxmox_auto_enroll.sh
        mode: '0700'

    - name: Run enrollment
```

```
      command: /root/proxmox_auto_enroll.sh
      environment:
        HOST_PREFIX: "{{ host_prefix }}"
        DEBUG: "true"
      register: enrollment_output

    - name: Show enrollment results
      debug:
        var: enrollment_output.stdout_lines
```

Advanced Playbook with Token Management

```
---
- name: Manage PatchMon Proxmox Integration
  hosts: localhost
  vars:
    patchmon_url: "https://patchmon.example.com"
    admin_token: "{{ vault_patchmon_admin_token }}"

  tasks:
    - name: Create Proxmox enrollment token
      uri:
        url: "{{ patchmon_url }}/api/v1/auto-enrollment/tokens"
        method: POST
        headers:
          Authorization: "Bearer {{ admin_token }}"
          Content-Type: "application/json"
        body_format: json
        body:
          token_name: "{{ inventory_hostname }}-proxmox"
          max_hosts_per_day: 200
          default_host_group_id: "{{ proxmox_host_group_id }}"
          allowed_ip_ranges: ["{{ proxmox_host_ip }}"]
          expires_at: "2026-12-31T23:59:59Z"
          metadata:
            integration_type: "proxmox-lxc"
            environment: "{{ environment }}"
        status_code: 201
      register: token_response
```

```
    - name: Store token credentials
      set_fact:
        enrollment_token_key: "{{ token_response.json.token.token_key }}"
        enrollment_token_secret: "{{ token_response.json.token.token_secret }}"

    - name: Deploy enrollment script to Proxmox hosts
      include_tasks: deploy_enrollment.yml
      vars:
        enrollment_token_key: "{{ enrollment_token_key }}"
        enrollment_token_secret: "{{ enrollment_token_secret }}"
```

Playbook for Bulk Enrollment via API

```
---
- name: Bulk enroll Proxmox containers
  hosts: proxmox_hosts
  become: yes
  vars:
    patchmon_url: "https://patchmon.example.com"
    token_key: "{{ vault_patchmon_token_key }}"
    token_secret: "{{ vault_patchmon_token_secret }}"

  tasks:
    - name: Get LXC container list
      shell: |
        pct list | tail -n +2 | while read -r line; do
          vmid=$(echo "$line" | awk '{print $1}')
          name=$(echo "$line" | awk '{print $3}')
          status=$(echo "$line" | awk '{print $2}')

          if [ "$status" = "running" ]; then
            machine_id=$(pct exec "$vmid" -- bash -c "cat /etc/machine-id 2>/dev/null || cat
/var/lib/dbus/machine-id 2>/dev/null || echo 'proxmox-lxc-$vmid-'$(cat
/proc/sys/kernel/random/uuid)" 2>/dev/null || echo "proxmox-lxc-$vmid-unknown")
            echo "{\"friendly_name\":\"$name\",\"machine_id\":\"$machine_id\"}"
          fi
        done | jq -s '.'
      register: containers_json
```

```
    - name: Bulk enroll containers
      uri:
        url: "{{ patchmon_url }}/api/v1/auto-enrollment/enroll/bulk"
        method: POST
        headers:
          X-Auto-Enrollment-Key: "{{ token_key }}"
          X-Auto-Enrollment-Secret: "{{ token_secret }}"
          Content-Type: "application/json"
        body_format: json
        body:
          hosts: "{{ containers_json.stdout | from_json }}"
        status_code: 201
      register: enrollment_result

    - name: Display enrollment results
      debug:
        msg: "{{ enrollment_result.json.message }}"
```

Ansible Role

```
# roles/patchmon_proxmox/tasks/main.yml
---
- name: Install PatchMon dependencies
  package:
    name:
      - curl
      - jq
    state: present

- name: Create PatchMon directory
  file:
    path: /opt/patchmon
    state: directory
    mode: '0755'

- name: Download enrollment script
  get_url:
    url: "{{ patchmon_url }}/api/v1/auto-enrollment/script?type=proxmox-lxc&token_key={{
```

```
token_key }}&token_secret={{ token_secret }}&force={{ force_install | default('false') }}"
    dest: /opt/patchmon/proxmox_auto_enroll.sh
    mode: '0700'

- name: Run enrollment script
  command: /opt/patchmon/proxmox_auto_enroll.sh
  environment:
    PATCHMON_URL: "{{ patchmon_url }}"
    AUTO_ENROLLMENT_KEY: "{{ token_key }}"
    AUTO_ENROLLMENT_SECRET: "{{ token_secret }}"
    HOST_PREFIX: "{{ host_prefix | default('') }}"
    DRY_RUN: "{{ dry_run | default('false') }}"
    DEBUG: "{{ debug | default('false') }}"
    FORCE_INSTALL: "{{ force_install | default('false') }}"
  register: enrollment_output

- name: Display enrollment results
  debug:
    var: enrollment_output.stdout_lines
  when: enrollment_output.stdout_lines is defined

- name: Fail if enrollment had errors
  fail:
    msg: "Enrollment failed with errors"
  when: enrollment_output.rc != 0
```

Ansible Vault for Credentials

```
# group_vars/all/vault.yml (encrypted with ansible-vault)
---
vault_patchmon_admin_token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
vault_patchmon_token_key: "patchmon_ae_abc123..."
vault_patchmon_token_secret: "def456ghi789..."
```

Playbook with Error Handling and Retries

```
---
- name: Robust Proxmox enrollment with error handling
```

```
  hosts: proxmox_hosts
  become: yes
  vars:
    patchmon_url: "https://patchmon.example.com"
    token_key: "{{ vault_patchmon_token_key }}"
    token_secret: "{{ vault_patchmon_token_secret }}"
    max_retries: 3
    retry_delay: 30

  tasks:
    - name: Test PatchMon connectivity
      uri:
        url: "{{ patchmon_url }}/api/v1/auto-enrollment/tokens"
        method: GET
        headers:
          Authorization: "Bearer {{ vault_patchmon_admin_token }}"
        status_code: 200
      retries: "{{ max_retries }}"
      delay: "{{ retry_delay }}"

    - name: Download enrollment script
      get_url:
        url: "{{ patchmon_url }}/api/v1/auto-enrollment/script?type=proxmox-lxc&token_key={{
token_key }}&token_secret={{ token_secret }}"
        dest: /root/proxmox_auto_enroll.sh
        mode: '0700'
      retries: "{{ max_retries }}"
      delay: "{{ retry_delay }}"

    - name: Run enrollment with retry logic
      shell: |
        for i in {1..{{ max_retries }}}; do
          echo "Attempt $i of {{ max_retries }}"
          if /root/proxmox_auto_enroll.sh; then
            echo "Enrollment successful"
            exit 0
          else
            echo "Enrollment failed, retrying in {{ retry_delay }} seconds..."
            sleep {{ retry_delay }}
```

```
          fi
        done
        echo "All enrollment attempts failed"
        exit 1
      register: enrollment_result

    - name: Handle enrollment failure
      fail:
        msg: "Proxmox enrollment failed after {{ max_retries }} attempts"
      when: enrollment_result.rc != 0

    - name: Parse enrollment results
      set_fact:
        enrolled_count: "{{ enrollment_result.stdout | regex_search('Successfully
Enrolled:\\s+(\\d+)', '\\1') | default('0') }}"
        failed_count: "{{ enrollment_result.stdout | regex_search('Failed:\\s+(\\d+)', '\\1')
| default('0') }}"

    - name: Report enrollment statistics
      debug:
        msg: |
          Enrollment completed:
          - Successfully enrolled: {{ enrolled_count }} containers
          - Failed: {{ failed_count }} containers
```

Error Handling

HTTP Status Codes

Code	Meaning	When It Occurs
`200`	OK	Successful read/update operations
`201`	Created	Token or host created successfully
`400`	Bad Request	Validation errors, invalid host group, invalid script type
`401`	Unauthorized	Missing, invalid, or expired credentials
`403`	Forbidden	IP address not in token's whitelist
`404`	Not Found	Token or resource not found

Code	Meaning	When It Occurs
`429`	Too Many Requests	Token's daily host creation limit exceeded
`500`	Internal Server Error	Unexpected server error

Error Response Formats

Simple error:

```
{
  "error": "Error message describing what went wrong"
}
```

Error with detail:

```
{
  "error": "Rate limit exceeded",
  "message": "Maximum 100 hosts per day allowed for this token"
}
```

Validation errors (400):

```
{
  "errors": [
    {
      "msg": "Token name is required (max 255 characters)",
      "param": "token_name",
      "location": "body"
    }
  ]
}
```

Rate Limiting

Token-Based Rate Limits

Each auto-enrollment token has a configurable `max_hosts_per_day` limit:

- **Default**: 100 hosts per day per token

- **Range**: 1–1000 hosts per day
- **Reset**: Daily (when the first request of a new day is received)
- **Scope**: Per-token, not per-IP

When the limit is exceeded, the API returns `429 Too Many Requests`:

```
{
  "error": "Rate limit exceeded",
  "message": "Maximum 100 hosts per day allowed for this token"
}
```

For bulk enrollment, the remaining daily quota is checked against the request size. If the request contains more hosts than the remaining quota allows, the entire request is rejected:

```
{
  "error": "Rate limit exceeded",
  "message": "Only 5 hosts remaining in daily quota"
}
```

Global Rate Limiting

The auto-enrollment endpoints are also subject to the server's global authentication rate limiter, which applies to all authentication-related endpoints.

Security Considerations

Token Security

- **Secret hashing**: Token secrets are hashed with bcrypt (cost factor 10) before storage
- **One-time display**: Secrets are only returned during token creation
- **Rotation**: Recommended every 90 days
- **Scope limitation**: Tokens can only create hosts — they cannot read, modify, or delete existing host data

IP Restrictions

Tokens support IP whitelisting with both exact IPs and CIDR notation:

```
{
  "allowed_ip_ranges": ["192.168.1.10", "10.0.0.0/24"]
```

```
}
```

IPv4-mapped IPv6 addresses (e.g. `::ffff:192.168.1.10`) are automatically handled.

Host API Key Security

- Host API keys (`api_key`) are hashed with bcrypt before storage
- The installation script uses a bootstrap token mechanism — the actual API credentials are not embedded in the script
- Bootstrap tokens are single-use and expire after 5 minutes

Network Security

- Always use HTTPS in production
- The `ignore_ssl_self_signed` server setting automatically configures curl flags in served scripts
- Implement firewall rules to restrict PatchMon server access to known IPs

Audit Trail

All enrollment activity is logged:

- Token name included in host notes (e.g. "Auto-enrolled via Production Proxmox on 2025-10-11T14:30:00Z")
- Token creation tracks `created_by_user_id`
- `last_used_at` timestamp updated on each enrollment

Complete Endpoint Summary

Admin Endpoints (JWT Authentication)

Method	Path	Description
`POST`	`/api/v1/auto-enrollment/tokens`	Create token
`GET`	`/api/v1/auto-enrollment/tokens`	List all tokens
`GET`	`/api/v1/auto-enrollment/tokens/{tokenId}`	Get single token
`PATCH`	`/api/v1/auto-enrollment/tokens/{tokenId}`	Update token
`DELETE`	`/api/v1/auto-enrollment/tokens/{tokenId}`	Delete token

Enrollment Endpoints (Token Authentication)

Method	Path	Description
`GET`	`/api/v1/auto-enrollment/script?type=...`	Download enrollment script
`POST`	`/api/v1/auto-enrollment/enroll`	Enroll single host
`POST`	`/api/v1/auto-enrollment/enroll/bulk`	Bulk enroll hosts (max 50)

Host Endpoints (API Credentials)

Method	Path	Description
`GET`	`/api/v1/hosts/install`	Download installation script
`GET`	`/api/v1/hosts/agent/download`	Download agent binary/script
`POST`	`/api/v1/hosts/update`	Report host data

Quick Reference: curl Examples

Create a token:

```
curl -X POST \
  -H "Authorization: Bearer <jwt_token>" \
  -H "Content-Type: application/json" \
  -d '{
    "token_name": "Production Proxmox",
    "max_hosts_per_day": 100,
    "default_host_group_id": "uuid",
    "allowed_ip_ranges": ["192.168.1.10"]
  }' \
  https://patchmon.example.com/api/v1/auto-enrollment/tokens
```

Download and run enrollment script:

```
curl -s "https://patchmon.example.com/api/v1/auto-enrollment/script?type=proxmox-lxc&token_key=KEY&token_secret=SECRET" | bash
```

Enroll a host directly:

```
curl -X POST \
  -H "X-Auto-Enrollment-Key: patchmon_ae_abc123..." \
```

```
  -H "X-Auto-Enrollment-Secret: def456ghi789..." \
  -H "Content-Type: application/json" \
  -d '{
    "friendly_name": "webserver",
    "machine_id": "proxmox-lxc-100-abc123"
  }' \
  https://patchmon.example.com/api/v1/auto-enrollment/enroll
```

Download agent installation script:

```
curl -H "X-API-ID: patchmon_abc123" \
     -H "X-API-KEY: def456ghi789" \
     https://patchmon.example.com/api/v1/hosts/install | bash
```

Integration Patterns

Pattern 1: Script-Based (Simplest)

```
# Download and execute in one command — credentials are injected into the script
curl -s "https://patchmon.example.com/api/v1/auto-enrollment/script?type=proxmox-lxc&token_key=KEY&token_secret=SECRET" | bash
```

Pattern 2: API-First (Most Control)

```
# 1. Create token via admin API
# 2. Enroll hosts via enrollment API (single or bulk)
# 3. Download agent scripts using per-host API credentials
# 4. Install agents with host-specific credentials
```

Pattern 3: Hybrid (Recommended for Automation)

```
# 1. Create token via admin API (or UI)
# 2. Download enrollment script with token embedded
# 3. Distribute and run script on Proxmox hosts
# 4. Script handles both enrollment and agent installation
```

Revision #3
Created 2025-10-14 17:53:17 UTC by Iby
Updated 2026-02-12 01:07:49 UTC by Iby