INVENQOR AGENT · OPERATIONS & SECURITY

관리자 가이드

수집 데이터 사전, 배포, 인증, 운영 통제, 모니터링과 장애 대응 기준서

대상 버전	Agent v0.2.3 · Server v0.2.5
문서 버전	1.0
기준일	2026-07-29
문서 등급	공개

> Server v0.2.5 운영자는 Server 설치 및 운영 가이드를 먼저 확인하십시오. 문서에는 PostgreSQL/SQLite 선택, 최초 관리자, Agent Bearer·mTLS 등록, 장애 spool과 Kubernetes 배포가 포함됩니다.

중앙 Server 운영 핵심

- 관리 콘솔은 자산·원천·변경 이력, 관계 그래프, Agent, Query DSL, 설정 버전, 감사 로그를 역할 권한에 따라 제공합니다.
- PostgreSQL은 운영 Primary이고 SQLite는 기동 시 연결 실패에만 사용하는 대체 모드입니다. 운영 중 PostgreSQL 장애에서는 SQLite로 전환하지 않습니다.
- 비밀 설정은 AES-256-GCM으로 암호화되고 API에는 구성 여부만 표시됩니다.
- 로컬 인증은 Argon2id, 계정 잠금, TOTP와 Recovery Code를 지원하며 Keycloak은 Authorization Code+PKCE, State, Nonce와 Role/Group Mapping을 검증합니다.
- Event ID는 Agent별 멱등 키입니다. Collector 오류로 삭제를 추론하지 않고 `removed` 변경만 논리 삭제합니다.
- CMDB 자동화와 AI Agent는 사람 Session을 공유하지 않고 scope 기반 API key와 stateless `/mcp`를 사용합니다. 자세한 수명주기는 자산 API·MCP·키 관리 가이드를 따릅니다.

문서 범위와 독자

이 문서는 Invenqor Agent v0.2.3를 운영 환경에 배포하는 Linux, 보안, 네트워크, CMDB/게이트웨이 관리자를 위한 기준서입니다. 다음 범위를 다룹니다.

- 지원 환경, 패키지 무결성 검증과 init 시스템별 설치
- 모든 설정 키와 인증 방식
- 수집 레코드의 원천, 필드, 식별자, 개인정보 경계와 제한
- 스냅샷, 변경 이벤트, 하트비트, 로컬 큐와 재시도 동작
- 게이트웨이 계약, 파일 권한, 모니터링, 업그레이드, 롤백과 장애 대응

v0.2.3에는 중앙 Server·대시보드·서명 자동 업데이트·자산 API·MCP가 포함됩니다. CVE 매핑, 자동 시정 정책 엔진과 원격 명령은 포함되지 않습니다.

1. 운영 아키텍처

```
Linux 호스트
  ├── 읽기: /proc, /sys, /etc, 패키지 DB, init 상태
  ├── Invenqor Agent (비특권 전용 계정)
  ├── 상태: /var/lib/invenqor-agent (0700)
  │     └── queue/*.jsonl (0600, 승인 전 삭제 금지)
  └── outbound HTTPS
        └── POST {server.url}/v1/agent/events
              └── Invenqor Server :7070 / PostgreSQL / CMDB · AI 연계
```

핵심 설계 원칙:

1. **비특권 실행**: root 권한과 Linux capability 없이 동작합니다.
2. **점진적 기능 저하**: 수집기 하나가 실패해도 나머지는 계속 수집합니다.
3. **보수적 삭제 판정**: 수집 오류가 있으면 누락을 자산 삭제로 만들지 않습니다.
4. **at-least-once 전달**: 게이트웨이 승인 전 큐 파일을 삭제하지 않습니다.
5. **outbound-only**: 에이전트가 수신 포트나 원격 셸을 열지 않습니다.
6. **최소 외부 명령**: `rpm`, `systemctl`, `rc-status` 만 고정 인자로 호출합니다.

2. 지원 기준과 사전 조건

2.1 플랫폼 기준

등급	대상	기대 범위
정식 목표	Kernel 3.10+, RHEL 계열 7+, Ubuntu 18.04+, Debian 10+	전체 기본 수집
호환 목표	Alpine, Amazon Linux, SUSE	핵심 수집, init별 차이 허용
제한 목표	Kernel 2.6 계열, CentOS 6 등	`/proc` 기반 핵심 수집
미지원	Kernel 2.4, Linux 이외 Unix	실행 보장 없음

제공 아키텍처:

- `x86_64-unknown-linux-musl`, CPU 기준 `x86-64`
- `aarch64-unknown-linux-musl`, CPU 기준 `generic`

두 바이너리는 외부 glibc에 의존하지 않는 정적 링크 결과물입니다. 정적 링크는 오래된 커널의 시스템 호출 호환성을 자동으로 보장하지 않습니다.

2.2 배포 전 체크리스트

- [ ] 대상 호스트의 `uname -m`과 패키지 아키텍처 일치
- [ ] 릴리즈 아카이브의 SHA-256 검증
- [ ] 조직 승인 게이트웨이 URL과 DNS 준비
- [ ] 단일 TCP 7070 outbound 및 프록시/방화벽 정책 확인
- [ ] 장비별 bearer token 또는 mTLS 인증서 발급
- [ ] 사설 CA 사용 시 CA PEM 배포
- [ ] 자산 데이터의 등급, 보존 기간, 접근 권한 확정
- [ ] 프로세스 명령행 수집은 기본 비활성 확인
- [ ] 시험 호스트에서 수집·전송·장애 복구 검증
- [ ] 설치/업그레이드/롤백 변경 승인

3. 패키지 검증과 설치

3.1 릴리즈 다운로드

x86_64 예시:

```
curl -fLO https://github.com/hkjang/invenqor-agents/releases/download/v0.2.3/invenqor-agent-linux-x86_64.tar.gz
curl -fLO https://github.com/hkjang/invenqor-agents/releases/download/v0.2.3/invenqor-agent-linux-
x86_64.tar.gz.sha256
sha256sum -c invenqor-agent-linux-x86_64.tar.gz.sha256
```

검증 결과가 `OK`가 아니면 배포를 중단합니다. SHA-256은 전송 오류와 변조 탐지에 사용하지만, v0.2.3는 별도 서명 파일이나 공급망 증명(attestation)을 제공하지 않습니다. 고통제 환경에서는 승인된 내부 저장소로 반입한 뒤 조직 서명을 추가하십시오.

3.2 패키지 구성

경로	설치 대상	모드
`bin/invenqor-agent`	`/opt/invenqor-agent/bin/invenqor-agent`	`0755`
`config/config.toml`	`/etc/invenqor-agent/config.toml`	`0640`, `root:invenqor-agent`
`README.md`	`/opt/invenqor-agent/README.md`	`0644`
init 정의	init 시스템별 시스템 경로	`0644` 또는 `0755`
상태 디렉터리	`/var/lib/invenqor-agent`	`0700`, 전용 계정

설치:

```
tar -xzf invenqor-agent-linux-x86_64.tar.gz
cd invenqor-agent-linux-x86_64
sudo ./scripts/install.sh
```

설치 스크립트는 `invenqor-agent` 시스템 사용자/그룹을 만들고, 기존 `/etc/invenqor-agent/config.toml` 은 덮어쓰지 않으며, 감지한 init 시스템에 서비스를 등록하고 시작합니다.

3.3 init 시스템별 설치 결과

systemd

- 서비스 파일: `/etc/systemd/system/invenqor-agent.service`
- 자동 시작: `systemctl enable --now invenqor-agent.service`
- 실행 사용자: `invenqor-agent:invenqor-agent`
- 쓰기 허용 경로: `/var/lib/invenqor-agent`

```
sudo systemctl daemon-reload
sudo systemctl enable --now invenqor-agent
sudo systemctl status invenqor-agent --no-pager
```

OpenRC

- 서비스 파일: `/etc/init.d/invenqor-agent`
- 기본 runlevel에 등록
- 로그: `/var/log/invenqor-agent.log`

```
sudo rc-update add invenqor-agent default
sudo rc-service invenqor-agent start
sudo rc-service invenqor-agent status
```

SysV init

- 서비스 파일: `/etc/init.d/invenqor-agent`
- `update-rc.d` 또는 `chkconfig` 로 등록

```
sudo service invenqor-agent start
sudo service invenqor-agent status
```

3.4 수동 설치 시 주의

조직 패키징 도구로 재작성할 수 있지만 다음 불변 조건을 유지해야 합니다.

- 전용 비로그인 계정 사용
- 설정과 상태 디렉터리를 일반 사용자가 읽거나 쓰지 못하게 제한

- `agent-id`, 큐, 인벤토리 파일의 `0600` 보장
- state directory와 queue directory의 `0700` 보장
- release 빌드에서 HTTPS만 허용
- 서비스 중복 실행 방지

4. 설정 기준서

4.1 전체 예시

```
[server]
url = "https://inventory.example.internal:7070"
enrollment_token_file = "/etc/invenqor-agent/enrollment.token"
# ca_file = "/etc/invenqor-agent/ca.pem"
# client_identity_pem = "/etc/invenqor-agent/device.pem"
allow_insecure_http = false
timeout_seconds = 30

[agent]
state_dir = "/var/lib/invenqor-agent"
interval_seconds = 900
heartbeat_seconds = 300
max_backoff_seconds = 3600
max_queue_bytes = 104857600

[updates]
enabled = true
channel = "stable"
check_interval_seconds = 21600
public_key = "base64-ed25519-public-key"
install_path = "/opt/invenqor-agent/bin/invenqor-agent"

[collectors]
os = true
cpu = true
memory = true
disk = true
network = true
process = true
packages = true
services = true
accounts = true
containers = true
include_process_cmdline = false
max_processes = 10000
```

설정 스키마는 알 수 없는 필드를 거부합니다. 오타가 묵인되지 않으므로 배포 전에 반드시 검증하십시오.

```
sudo -u invenqor-agent \
  /opt/invenqor-agent/bin/invenqor-agent \
  --config /etc/invenqor-agent/config.toml \
  --validate-config
```

4.2 설정 키

키	기본값	제약·의미
`server.url`	없음	Server 기본 URL. 운영은 단일 HTTPS `:7070` 사용
`server.allow_insecure_http`	`false`	공인 주소 HTTP의 명시 허용; 사설/내부 HTTP는 URL만으로 허용
`server.enrollment_token`	없음	Server가 보호 모드일 때만 필요한 공용 Token, 32자 이상
`server.enrollment_token_file`	없음	선택적 공용 등록 Token 파일의 절대 경로
`server.bearer_token`	없음	수동 등록 예외 장비의 장비별 bearer token
`server.ca_file`	없음	사설 루트 CA PEM 경로
`server.client_identity_pem`	없음	클라이언트 인증서 체인+개인키 PEM
`server.timeout_seconds`	`30`	전체 HTTP 요청 제한, 0 금지
`agent.state_dir`	`/var/lib/invenqor-agent`	ID, 인벤토리, 해시, 큐 저장
`agent.interval_seconds`	`900`	전체 수집 주기, 0 금지
`agent.heartbeat_seconds`	`300`	변경이 없을 때 생존 이벤트 주기, 0 금지
`agent.max_backoff_seconds`	`3600`	전송 재시도 상한. 0이면 내부적으로 최소 1초
`agent.max_queue_bytes`	`104857600`	큐 전체 바이트 한도
`updates.enabled`	`false`	서명 업데이트 주기 확인 활성화
`updates.channel`	`stable`	`stable` 또는 `beta`
`updates.check_interval_seconds`	`21600`	확인 주기, 최소 300초
`updates.public_key`	없음	base64 Ed25519 공개키, 활성화 시 필수
`updates.install_path`	`/opt/...`	root helper가 교체할 절대 경로
`collectors.<name>`	`true`	개별 수집기 활성화
`collectors.include_process_cmdline`	`false`	프로세스 argv 포함 여부
`collectors.max_processes`	`10000`	PID 정렬 후 수집 상한, 0 금지

4.3 인증 선택

장비별 bearer token

장점은 배포 단순성이고, 단점은 토큰 파일 유출 시 재사용 위험입니다.

- 토큰은 호스트별로 발급합니다.
- 중앙 로그와 지원 티켓에서 토큰을 마스킹합니다.
- 회전 시 새 토큰 배포 → 설정 검증 → 재시작 → 수신 확인 → 구 토큰 폐기 순서를 사용합니다.
- 설정 파일을 `root:invenqor-agent`, `0640` 이하로 제한합니다.

mTLS

mTLS는 장비 인증과 전송 암호화를 결합합니다.

- `client_identity_pem`은 인증서 체인과 개인키가 함께 있는 PEM입니다.
- 에이전트 실행 사용자가 파일을 읽을 수 있어야 합니다.
- 만료 모니터링과 폐기 목록/OCSP 정책은 게이트웨이 운영 범위입니다.
- 사설 CA를 쓰면 `ca_file`에 신뢰 루트를 지정합니다.
- 개인키 PEM은 백업, 배포 로그, 티켓 첨부에서 제외합니다.

bearer token과 mTLS를 동시에 구성할 수 있습니다. 게이트웨이가 두 조건을 모두 요구하도록 설계했을 때만 함께 사용하십시오.

4.4 로그 수준

환경변수 `RUST_LOG`로 조정합니다. 기본은 `invenqor_agent=info`입니다.

systemd override 예시:

```
sudo systemctl edit invenqor-agent
```

```
[Service]
Environment=RUST_LOG=invenqor_agent=debug
```

```
sudo systemctl daemon-reload
sudo systemctl restart invenqor-agent
```

디버그 수준은 장애 분석 기간에만 사용하고 종료 후 원복합니다. 구현은 토큰, 인증서 원문, 프로세스 명령행을 직접 로그하지 않지만, 수집 결과를 `--once`로 출력하면 민감 자산정보가 노출될 수 있습니다.

5. 공통 데이터 모델

모든 자산 레코드는 다음 공통 필드를 가집니다.

필드	형식	설명
`asset_id`	string	범주별 안정 식별자
`category`	string	`system`, `process`, `software.package` 등의 범주
`source`	string	`/proc` 경로 또는 고정 외부 명령 등 데이터 원천
`collected_at`	Unix seconds	이 수집 주기의 기준 시각
`payload`	object	범주별 필드

스냅샷:

필드	설명
`schema_version`	현재 `1`
`agent_id`	상태 디렉터리에 생성한 UUID
`collected_at`	수집 시작 기준 Unix seconds
`duration_ms`	전체 수집 소요 시간
`records`	`asset_id`로 정렬한 레코드
`errors`	수집기명과 오류 메시지

5.1 자산 식별자 규칙

범주	`asset_id` 구성
프로세스	`process:{pid}:{start_ticks}`
패키지	`package:{manager}:{name}:{architecture}`
서비스	`service:{manager}:{name}`
파일시스템	`filesystem:{mountpoint}`
네트워크 인터페이스	`interface:{name}`
사용자 계정	`user:{uid}:{name}`
단일 레코드 범주	범주명 자체

프로세스는 PID 재사용을 구분하기 위해 커널 시작 tick을 함께 사용합니다.

6. 수집 데이터 상세 사전

6.1 시스템 (`system`)

원천: `/etc/os-release` 또는 `/usr/lib/os-release` , `/proc/sys/kernel/*` , `/proc/stat` , `/etc/timezone` , `/etc/localtime`

필드	설명
`hostname`	커널 호스트명, 실패 시 `/etc/hostname` , 최종 `unknown`
`architecture`	빌드 대상 아키텍처
`kernel_release`	커널 릴리즈
`kernel_version`	커널 빌드 버전 문자열
`boot_time`	`/proc/stat` 의 `btime` , Unix seconds
`timezone`	`/etc/timezone` 또는 zoneinfo 심볼릭 링크
`os_release`	`/etc/os-release` 의 대문자 키를 소문자로 변환한 맵

제한: DMI 제조사·시리얼, BIOS, 메인보드 정보는 수집하지 않습니다. 에이전트는 로컬 ID 초기화 시 machine-id와 DMI product UUID를 읽어 info 로그에 기록할 수 있지만 v0.2.3 인벤토리 레코드나 전송 envelope에는 포함하지 않습니다.

6.2 CPU (`hardware.cpu`)

원천: `/proc/cpuinfo` , `/proc/loadavg`

필드	설명
`architecture`	실행 바이너리 아키텍처
`logical_cpus`	processor 블록 수
`physical_packages`	`physical id` 고유 수, 정보가 없고 CPU가 있으면 최소 1
`models`	고유 CPU 모델 문자열 목록
`load_average`	1분, 5분, 15분 load average

제한: 코어별 플래그, 주파수, 온도, 소켓/코어/스레드의 완전한 토폴로지는 수집하지 않습니다.

6.3 메모리 (`hardware.memory`)

원천: `/proc/meminfo`

`/proc/meminfo`의 숫자 키를 가능한 한 모두 수집합니다. 키는 snake_case로 변환하고 `_bytes`를 붙이며, `kB`는 1024배로 바이트 단위 변환합니다. 예:

- `mem_total_bytes`, `mem_free_bytes`, `mem_available_bytes`
- `buffers_bytes`, `cached_bytes`
- `swap_total_bytes`, `swap_free_bytes`
- `huge_pages_total_bytes`

커널별로 제공 키가 다르므로 중앙 스키마는 알 수 없는 메모리 키를 허용해야 합니다. `HugePages_Total`처럼 실제 단위가 페이지 개수인 무단위 값에도 현재 구현은 `_bytes` 접미사를 사용한다는 점을 분석 시 유의하십시오.

6.4 파일시스템 (`hardware.filesystem`)

원천: `/proc/self/mounts` 또는 `/proc/mounts`, `statvfs(2)`

필드	설명
`device`	마운트 원본 장치/경로
`mountpoint`	마운트 지점, asset ID 기준
`filesystem`	파일시스템 형식
`options`	마운트 옵션 배열
`usage.total_bytes`	전체 바이트
`usage.available_bytes`	비특권 사용자에게 사용 가능한 바이트
`usage.free_bytes`	전체 여유 바이트
`usage.files`	inode 총수
`usage.files_free`	여유 inode 수

제한: pseudo filesystem과 bind mount도 포함될 수 있습니다. `statvfs` 실패 시 해당 마운트의 `usage`는 null이지만 레코드는 유지됩니다.

6.5 네트워크 인터페이스 (`network.interface`)

원천: `/sys/class/net`, `getifaddrs(3)`

필드	설명
`name`	인터페이스명
`mac`	링크 주소
`state`	`operstate`
`mtu`	MTU

필드	설명
`ifindex`	커널 인터페이스 인덱스
`addresses`	IPv4/IPv6 주소 문자열 배열

네트워크 네임스페이스 기준으로 보이는 인터페이스만 수집합니다. 프리픽스 길이, 브로드캐스트, VLAN/본딩 관계는 v0.2.3에 포함되지 않습니다.

6.6 네트워크 구성 (`network.configuration`)

원천: `/proc/net/route` , `/proc/net/tcp*` , `/proc/net/udp*` , `/etc/resolv.conf`

필드	설명
`default_routes[]`	IPv4 기본 경로의 인터페이스, 게이트웨이, metric
`dns_servers[]`	`resolv.conf` 의 nameserver 값
`listening[]`	protocol, local address, local port

TCP는 상태 `LISTEN` 만 포함합니다. UDP는 연결 상태 개념 차이로 `/proc/net/udp*` 의 로컬 endpoint를 포함합니다. IPv6 주소는 수집하지만 v0.2.3의 기본 경로 수집은 IPv4 `/proc/net/route` 만 사용합니다. 소켓과 프로세스의 연결 관계는 제공하지 않습니다.

6.7 프로세스 (`process`)

원천: `/proc/[pid]/status` , `/proc/[pid]/stat` , `/proc/[pid]/exe` , 선택적으로 `/proc/[pid]/cmdline`

필드	설명
`pid`	프로세스 ID
`name`	커널 프로세스 이름
`state`	커널 상태 문자
`parent_pid`	부모 PID
`start_ticks`	부팅 이후 시작 tick, PID 재사용 구분
`uid` , `gid`	status의 첫 UID/GID
`executable`	실행 파일 심볼릭 링크 대상, 권한 거부 시 null
`cmdline`	기본 null, 활성화 시 argv 문자열 배열

PID를 숫자 순으로 정렬한 뒤 `max_processes` 까지만 처리합니다. 수집 도중 종료된 프로세스와 권한 때문에 읽지 못한 프로세스는 제외합니다. 환경변수, 열린 파일, 메모리 내용은 수집하지 않습니다.

> **개인정보/비밀정보 위험:** `include_process_cmdline=true`는 명령행에 포함된 비밀번호, 토큰, 사용자 입력, 파일 경로를 중앙으로 전송할 수 있습니다. 법무·보안 승인, 최소권한, 보존 기간, 마스킹 정책 없이 활성화하지 마십시오.

6.8 패키지 (`software.package`)

관리자	원천	필드
dpkg	`/var/lib/dpkg/status`	manager, name, version, architecture, source_package
apk	`/lib/apk/db/installed`	manager, name, version, architecture
rpm	고정 인자 `rpm -qa --qf ...`	manager, name, epoch/version/release, architecture

dpkg는 `install ok installed`만 포함합니다. RPM DB 형식 차이는 rpm 클라이언트를 호환 경계로 사용하며 셸을 거치지 않습니다. 여러 패키지 관리자가 공존할 때 dpkg → apk → rpm 우선순위로 하나만 선택합니다. 언어별 패키지(pip, npm 등), 컨테이너 이미지 내부 패키지, 파일 해시는 수집하지 않습니다.

6.9 서비스 (`service`)

관리자	원천	필드·제한
systemd	고정 인자 `systemctl show --all --type=service`	name, load/active/sub state, unit file state
OpenRC	`/etc/init.d`, `rc-status --all`	서비스명, 일치하는 상태 문자열
SysV	`/etc/init.d`, `/etc/rc*.d`	서비스명, 활성화 runlevel; 현재 실행 상태는 null

systemd 조회가 실패하면 OpenRC 또는 SysV로 점진적으로 대체합니다. 서비스 설정 본문, 환경변수, 자격증명은 수집하지 않습니다.

6.10 사용자 계정 (`account.user`)

원천: `/etc/passwd`, `/etc/group`

필드	설명
`name`	로그인명
`uid`, `gid`	숫자 사용자/기본 그룹 ID
`gecos`	설명/실명 필드일 수 있음
`home`	홈 디렉터리
`shell`	로그인 셸
`supplementary_groups`	`/etc/group`에 명시된 보조 그룹명

비밀번호 해시와 `/etc/shadow`는 읽지 않습니다. GECOS, 사용자명, 홈 경로는 개인정보가 될 수 있으므로 중앙 저장소 접근과 보존을 통제하십시오. LDAP/AD 등 NSS 외부 계정은 `/etc/passwd`에 정적으로 나타나지 않으면 포함되지 않습니다.

6.11 컨테이너 환경 (`container.environment`)

원천: 런타임 소켓 경로, `/proc/1/cgroup`, `/.dockerenv`, `/run/.containerenv`, cgroup filesystem, Kubernetes service account 경로

필드	설명
`host_runtime_endpoints`	발견한 docker/containerd/podman/crio 소켓
`agent_is_containerized`	cgroup/표식 기반 에이전트 컨테이너 실행 여부
`cgroup_version`	cgroup v1 또는 v2
`kubernetes_service_account`	서비스 계정 디렉터리 존재 여부

컨테이너, Pod, 이미지, 레지스트리, Kubernetes 리소스를 열거하지 않습니다. 런타임 소켓에 접속하지 않고 존재 여부만 봅니다.

7. 수집 실패와 변경 판정

수집기는 병렬 blocking task로 실행되고 결과와 오류는 이름순/ID순으로 정렬됩니다. 한 수집기가 실패하면 `errors[]`와 전송 envelope의 `collection_errors[]`에 기록되고 나머지 수집 결과는 유지됩니다.

변경 판정은 `collected_at`, 전체 수집 소요 시간처럼 매번 바뀌는 값은 제외하고, 각 레코드의 ID, 범주, 원천, payload를 SHA-256으로 계산합니다.

상황	전송 내용
최초 수집	전체 `snapshot`
후속 변경	`added`, `updated`, `removed` change 배열
변경 없음, heartbeat 도래	snapshot/change 없는 heartbeat
수집기 오류	오류 포함, 누락 자산의 제거 판정 억제

수집기 오류가 하나라도 있으면 그 주기는 **전체 인벤토리의 제거 판정**을 보수적으로 억제합니다. 이는 오탐 대량 삭제를 막지만, 다른 정상 수집기에서 실제로 사라진 자산의 제거 이벤트도 다음 완전 성공 주기까지 지연할 수 있습니다.

8. 상태 저장과 내구성 큐

기본 루트는 `/var/lib/invenqor-agent`입니다.

경로	내용	권한
`agent-id`	설치 인스턴스 UUID	`0600`
`inventory.json`	직전 유효 인벤토리	`0600`
`snapshot.sha256`	안정 스냅샷 해시	`0600`
`last-heartbeat`	마지막 이벤트 시각	`0600`
`queue/*.jsonl`	순서화된 미승인 envelope	`0600`

모든 상태 파일은 임시 파일 생성 → `fsync` → rename 방식으로 원자 교체합니다. 큐 파일명은 nanosecond 기반 순서값과 event UUID로 구성되어 동일 초에 생성해도 순서를 보존합니다. 전송은 파일명 순서대로 직렬 처리합니다.

큐 크기가 `max_queue_bytes`를 넘게 될 경우:

1. 기존 미전송 이벤트는 삭제하지 않습니다.
2. 새 envelope 생성을 실패시켜 데이터 손실을 명시적으로 드러냅니다.
3. 운영자는 원인 복구 후 큐가 정상 감소하는지 확인해야 합니다.

> 큐 파일을 수동 수정·재정렬·삭제하지 마십시오. 불가피한 폐기는 변경 승인, 백업, 영향 이벤트 범위, 중앙 수신 상태를 기록한 후 수행하십시오.

9. 게이트웨이 연동 계약

9.1 요청

```
POST {server.url}/v1/agent/events
Content-Type: application/json
User-Agent: invenqor-agent/0.2.3
X-Invenqor-Agent-Id: <agent UUID>
X-Invenqor-Event-Id: <event UUID>
Authorization: Bearer <token>   # 구성한 경우
```

주요 envelope 필드:

필드	설명
`schema_version`	현재 1
`event_id`	재시도 중 변하지 않는 idempotency key
`agent_id`	설치 인스턴스 UUID

필드	설명
`created_at`	이벤트 생성 Unix seconds
`kind`	`inventory` 또는 `heartbeat`
`snapshot_hash`	유효 인벤토리 SHA-256
`snapshot`	최초 인벤토리일 때 전체 스냅샷
`changes`	후속 added/updated/removed
`collection_errors`	해당 수집 주기 오류

9.2 성공 응답

```
{
  "accepted": true,
  "policy_version": "2026-07-29.1"
}
```

HTTP 2xx와 `accepted: true`가 모두 충족돼야 성공입니다. `policy_version`은 로그에 관찰만 하며 v0.2.3는 원격 정책이나 명령을 실행하지 않습니다.

게이트웨이는 `event_id`에 대해 멱등 처리해야 합니다. 네트워크 단절로 서버가 처리 후 응답을 보내지 못하면 같은 event가 재전송될 수 있습니다.

9.3 실패와 백오프

- DNS, TCP, TLS, 타임아웃, 비-2xx, JSON 오류, `accepted: false`는 실패
- 첫 재시도 대기 1초
- 실패할 때마다 2배 증가
- `max_backoff_seconds` 상한 적용
- 성공 시 1초로 초기화

10. 보안 통제

10.1 systemd sandbox

기본 unit은 다음을 적용합니다.

- `NoNewPrivileges=true`
- `ProtectSystem=strict`, `ProtectHome=true`, `PrivateTmp=true`
- 커널 tunable/module/control group 보호
- SUID/SGID, realtime, personality 변경 제한

- `MemoryDenyWriteExecute=true`
- capability와 ambient capability 없음
- 주소 패밀리 `AF_UNIX`, `AF_INET`, `AF_INET6`만 허용
- `/var/lib/invenqor-agent`만 쓰기 허용
- `UMask=0077`

조직 override가 이 통제를 약화시키지 않는지 구성 감사에 포함하십시오.

10.2 데이터 분류 권고

데이터	권고 등급 근거
호스트명, IP, 패키지, 서비스	보안 구조와 공격 표면 노출 가능
계정명, GECOS, 홈 경로	개인정보 또는 조직 식별정보 가능
프로세스 실행 파일	업무·보안 도구 구성 유추 가능
프로세스 명령행(선택)	자격증명·개인 데이터 포함 가능
mTLS 개인키/bearer token	인증 비밀, 인벤토리와 분리 관리

최종 등급은 조직 정책에 따릅니다. 중앙 게이트웨이는 전송 암호화뿐 아니라 저장 암호화, 역할 기반 접근, 조회 감사, 보존/파기 정책을 구현해야 합니다.

10.3 보안 경계와 잔여 위험

- 아카이브 SHA-256은 제공하지만 서명, SBOM, provenance는 없음
- URL-only 자동 등록은 7070에 도달 가능한 장비의 최초 등록을 허용하므로 IP/CIDR allowlist로 신뢰 대역을 제한하고, 경계 밖에 노출할 때 enrollment token 보호 모드 또는 자동 등록 비활성화 필요
- 보호 모드의 enrollment token은 최초 등록 권한이므로 Secret 관리와 정기 회전 필요
- 자동 등록 Agent는 장비 Token 무효화 시 device claim으로 자동 복구하지만, mTLS 인증서 발급·폐기는 조직 PKI 수명주기를 따름
- 로컬 큐 자체 암호화 없음(파일시스템 접근 통제에 의존)
- 중앙 Server의 RBAC·감사 로그를 적용하고 정기 접근권한 검토 필요
- 자동 업데이트는 서명 검증·원자 교체를 제공하지만 조직 승인과 rollback artifact 운영은 별도 절차 필요
- 원격 명령 기능 없음

11. 모니터링과 운영 지표

11.1 호스트 지표

최소 경보 항목:

지표	권고 기준
서비스 생존	5분 이상 inactive 시 경보
마지막 성공 수집	`interval_seconds` 의 2배 이상 지연 시 조사
큐 크기	한도의 70% 경고, 90% 긴급
큐 최장 event age	조직의 자산 최신성 SLO 초과 시 경보
반복 collection error	같은 수집기 3회 이상 연속 실패 시 조사
인증서 만료	30/14/7일 단계 경보

현재 에이전트는 Prometheus endpoint를 열지 않습니다. systemd 상태, 로그, 상태 디렉터리와 게이트웨이 수신 시각을 기존 모니터링에 연계하십시오.

11.2 명령 예시

```
systemctl is-active invenqor-agent
journalctl -u invenqor-agent --since "30 minutes ago" --no-pager
du -sb /var/lib/invenqor-agent/queue
find /var/lib/invenqor-agent/queue -type f -name '*.jsonl' | wc -l
```

큐의 가장 오래된 파일:

```
sudo find /var/lib/invenqor-agent/queue -maxdepth 1 \
  -type f -name '*.jsonl' -printf '%T@ %p\n' \
  | sort -n | head -1
```

11.3 중앙 지표

- 등록 agent 수와 최근 heartbeat agent 비율
- agent별 마지막 inventory/heartbeat 시각
- event 중복 제거 횟수
- HTTP/TLS/auth 거부율
- collection error 비율과 수집기별 상위 원인
- 이벤트 처리 지연 p50/p95/p99
- 스키마 버전 분포

11.1 멀티 Pod Server 진단 로그

`audit.read` 권한 사용자는 **Server 로그**에서 모든 Pod가 공용 PostgreSQL에 기록한 구조화 진단 이벤트를 조회합니다. Load Balancer가 어느 Pod로 화면 요청을 보내도 결과는 같으며 다음 필터를 제공합니다.

- `error`, `warning`, `info` 수준
- Agent 등록, Agent 전송, Server HTTP 구성요소
- Pod/instance ID
- request ID, Agent ID, 오류 코드, source IP 검색
- 15초 자동 갱신과 100/200/500건 조회

DB에는 일반 성공 access log나 원문 인벤토리를 복제하지 않습니다. Agent 등록 성공, 정책 거부, 인증·schema·처리 실패와 Server 내부 오류처럼 조사에 필요한 이벤트만 저장합니다. Token, Secret, Authorization, URL password는 기록 전에 redaction합니다. 기본 보존은 30일 또는 최신 10,000건 중 먼저 도달하는 한도이며, 컨테이너 stdout 로그는 조직의 중앙 로그 플랫폼으로 별도 수집하십시오.

Agent가 출력한 `request_id=...`를 화면 검색창에 붙여 넣으면 같은 요청을 처리한 Pod, 판정 source IP, 정책 버전과 실패 단계를 확인할 수 있습니다. Server API는 `GET /api/v1/admin/diagnostics/logs`이며 `level`, `component`, `instance_id`, `q`, `limit` 필터를 지원합니다.

12. 운영 절차

12.1 설정 변경

1. 설정 백업과 변경 승인 번호 기록
2. 비밀값이 로그/배포 결과에 노출되지 않게 배포
3. `--validate-config` 수행
4. 서비스 재시작
5. 로그에서 수집과 전송 확인
6. 게이트웨이에서 해당 agent의 event 수신 확인
7. 백업 보존 기한 후 안전 삭제

Server의 Agent 등록 정책은 예외적으로 재기동 없이 적용됩니다. 관리 콘솔 **설정 → Agent 등록**에서 `disabled`, Token 없는 URL-only `open`, 등록 Token이 필요한 `token` 모드를 선택합니다.

- `open`: Agent의 `[server].url`만 설정하면 최초 요청에서 자동 등록
- `token`: 화면에서 발급한 `ivq_et_...`를 Agent의 `enrollment_token` 또는 `enrollment_token_file`에 배포한 뒤 등록
- `disabled`: 신규 등록만 차단하고 기존 Agent 수집은 유지

인증 모드와 별도로 **접속 IP 정책**을 `모든 IP` 또는 `지정 IP만 허용`으로 설정합니다. 단일 주소와 CIDR을 한 줄에 하나씩 입력하고, Ingress/LB 뒤에서는 해당 프록시의 실제 peer 주소만 **신뢰 프록시**에 추가합니다. 신뢰 프록시가 아닌 접속의 `X-Forwarded-For`는 무시되므로 헤더 위조로 allowlist를 우회할 수 없습니다. 정책 변경 후 허용 대역의 canary Agent와 허용되지 않은 대역을 각각 시험하십시오.

등록 성공 시 자산 목록에는 수집 주기를 기다리지 않고 `discovered` host와 접속 IP 식별자가 나타납니다. 첫 system inventory 후 같은 자산이 `active`로 승격되어야 합니다. 동일 Agent UUID에 host가 두 개 생기거나 `discovered`가 계속 유지되면 Agent 이벤트 실패, claim 충돌, DB transaction 오류를 감사 로그와 `agent_events.processing_error`에서 확인합니다.

발급/회전 Token 원문은 한 번만 보이며 DB에는 SHA-256 비교값만 저장됩니다. Token 폐기 시 자동 등록이 활성 상태면 Open 모드가 됩니다. 정책과 버전은 공용 DB에 저장되고 등록 요청마다 읽으므로 Kubernetes 모든 Pod에 즉시 동일하게 적용됩니다. `AGENT_AUTO_ENROLLMENT`와 등록 Token 환경변수는 DB 정책이 없는 최초 기동의 초기값으로만 사용됩니다.

12.2 업그레이드

v0.2.3는 관리자가 승인한 Ed25519 서명 artifact의 자동 스테이징과 systemd root helper 기반 원자 교체를 지원합니다. 서명 개인키는 Server와 Agent에 배포하지 않고 오프라인 환경에서 보관합니다. Agent는 더 높은 버전, 일치하는 OS/Architecture, 128 MiB 이하 크기, SHA-256과 서명이 모두 맞을 때만 `pending.json`을 생성합니다. 기존 바이너리는 `.previous`로 보존됩니다.

1. 새 바이너리와 SHA-256, 변경 내역, 설정 호환성을 승인합니다.
2. 오프라인 Ed25519 키로 바이너리 원문을 서명합니다.
3. `agents.manage` 관리자 API로 stable/beta, 아키텍처와 rollout 비율을 게시합니다.
4. 1~10% canary에서 설치·수집·전송·재시작을 확인합니다.
5. 중앙 오류율과 `.previous` 생성 상태를 확인한 뒤 단계적으로 100% 확산합니다.
6. 실패하면 서비스를 중지하고 `.previous`를 원래 경로로 복원합니다.

상태 디렉터리와 `agent-id`를 유지해야 중앙에서 같은 장비로 연속 인식합니다.

12.3 롤백

1. 서비스 중지
2. 이전 검증 바이너리 복원
3. 설정/상태 스키마가 이전 버전과 호환되는지 확인
4. 이전 설정이 필요하면 승인된 백업 복원
5. 서비스 시작 후 큐와 중앙 수신 확인

큐를 삭제해 롤백 문제를 숨기지 마십시오. 새 버전이 만든 상태가 이전 버전과 호환되지 않는 경우 디렉터리 전체 백업 후 제품 담당자 판단을 받습니다.

12.4 제거

```
sudo ./scripts/uninstall.sh
```

스크립트는 서비스와 바이너리만 제거하고 설정과 상태/큐를 보존합니다. 완전 폐기:

1. 중앙 시스템에서 장비 폐기 승인
2. 미전송 큐 확인과 필요한 증적 보관
3. 인증서/token 폐기
4. 조직 보존 정책에 따라 설정·상태 안전 삭제
5. 서비스 계정 삭제 여부 결정
6. 자산관리대장 갱신

13. 장애 대응 런북

13.1 서비스 시작 실패

```
sudo systemctl status invenqor-agent --no-pager
sudo journalctl -u invenqor-agent -n 200 --no-pager
sudo -u invenqor-agent \
  /opt/invenqor-agent/bin/invenqor-agent \
  --config /etc/invenqor-agent/config.toml \
  --validate-config
sudo namei -l /etc/invenqor-agent/config.toml
sudo namei -l /var/lib/invenqor-agent
```

판단: TOML 오류 → 파일 권한 → CA/identity PEM → 상태 파일 손상 → 바이너리 아키텍처 순으로 확인합니다.

13.2 전송 실패

1. DNS와 443 outbound 확인
2. 시스템 시간/NTP 확인
3. 게이트웨이 인증서 SAN/만료/CA 확인
4. client identity 또는 token 유효성 확인
5. 게이트웨이 응답이 2xx JSON이며 `accepted: true`인지 확인
6. event ID 중복 처리가 정상인지 확인
7. 복구 후 오래된 큐부터 감소하는지 확인

에이전트 설정에 프록시 전용 키는 없습니다. reqwest가 따르는 환경 프록시 정책을 사용해야 한다면 서비스 환경과 조직 검증을 별도로 수행하십시오.

13.3 큐 포화

영향: 기존 이벤트는 보존되지만 새 이벤트 queueing이 실패합니다.

조치:

1. 서비스와 파일시스템 용량 확인
2. 전송 실패 원인 복구
3. 게이트웨이 수용량과 오류율 확인
4. 큐가 자연 감소하는지 관찰
5. 필요 시 승인 후 `max_queue_bytes` 확대와 디스크 여유 확보
6. event age와 누락 가능 시간대를 사고 기록에 남김

수동 폐기는 최후 수단입니다. 폐기 전 큐 백업, 해시, 시간 범위, event ID, 승인자를 남기십시오.

13.4 반복 수집기 오류

수집기	우선 확인
OS/CPU/메모리	`/proc` 마운트와 읽기 권한
디스크	`/proc/self/mounts`, 마운트 namespace, statvfs 권한
네트워크	`/sys/class/net`, `/proc/net`, namespace
프로세스	hidepid 등 `/proc` 정책, PID churn
패키지	DB 경로, rpm 실행 파일과 30초 제한
서비스	init 감지 경로, systemctl/rc-status와 15/10초 제한
계정	`/etc/passwd`, `/etc/group` 권한
컨테이너	namespace와 표식 경로

수집 오류가 있는 동안 삭제 이벤트가 보류될 수 있음을 중앙 데이터 사용자에게 알립니다.

13.5 상태 파일 손상

서비스를 중지하고 전체 상태 디렉터리를 읽기 전용 증적으로 복사한 뒤 조사합니다. `agent-id`를 새로 만들면 중앙에서는 새 agent로 인식합니다. `inventory.json`이나 해시를 제거하면 다음 수집이 최초 전체 snapshot처럼 전송될 수 있습니다. 임의 수정 전에 제품 담당자와 중앙 deduplication 영향을 검토하십시오.

14. 배포 자동화 권고

대규모 배포는 Ansible, Salt, Puppet, OS 패키지 저장소 등 조직 표준을 사용하고 다음 조건을 idempotent하게 구현합니다.

- 아키텍처에 맞는 아카이브 선택
- 체크섬 불일치 시 즉시 실패
- 전용 계정과 디렉터리 모드 보장
- 기존 설정 비파괴
- 비밀값이 명령행과 로그에 노출되지 않는 전달 방식
- 설정 검증 성공 후에만 서비스 재시작
- canary → 소규모 ring → 전체 순서
- 각 ring의 heartbeat와 queue SLO를 다음 단계 진입 조건으로 사용

15. 운영 인수 기준

다음 항목을 모두 증적화하면 운영 인수 완료로 판단할 수 있습니다.

- [ ] 대상 배포판/아키텍처별 설치 성공
- [ ] 재부팅 후 자동 시작
- [ ] 비특권 실행과 systemd sandbox 확인
- [ ] 설정/상태/인증서 파일 권한 확인
- [ ] 전체 기본 수집기 결과 또는 승인된 예외 기록
- [ ] 최초 full snapshot 수신
- [ ] 변경 event와 heartbeat 수신
- [ ] 네트워크 단절 중 큐 보존 및 복구 후 순차 전송
- [ ] 게이트웨이 idempotency 검증
- [ ] 잘못된 token/인증서 거부
- [ ] 큐 70/90%와 heartbeat 지연 경보
- [ ] 인증서/token 회전 절차
- [ ] 업그레이드·롤백 시험
- [ ] 제거 후 설정/상태 보존 동작 확인
- [ ] 데이터 분류, 접근, 보존, 파기 승인

16. 중앙 인증·사용자 운영

Server 관리 콘솔의 **설정 → Keycloak**은 OIDC Issuer/Realm, confidential client, Redirect URI, Scope, claim, Email domain, Role/Group mapping과 사설 CA를 관리합니다. 연결 테스트는 실제 discovery와 TLS 신뢰를 확인하며 Client Secret은 Master Key로 암호화되어 구성 여부만 표시됩니다. Client Secret 없이 SSO를 활성화하거나 존재하지 않는 내부 역할을 mapping하는 설정은 거부됩니다.

일반 구성은 **최소 정보 빠른 연동**에서 Keycloak 주소, Realm, Client ID, Client Secret만 입력합니다. InvenQor 외부 주소는 현재 브라우저 origin으로 채워지며 Ingress 외부 URL이 다를 때만 수정합니다. Server는 OIDC Discovery와 TLS 신뢰를 먼저 확인하고 Callback/Logout URI, 표준 scope·claim을 생성한 뒤 SSO를 활성화합니다. 기존 Secret이 있으면 재입력 없이 재검증할 수 있습니다. Discovery 실패 시 기존 운영 설정은 유지됩니다.

권장 Keycloak claim 구성:

목적	Claim 예시	Invenqor 입력
사용자 ID	`preferred_username`	Username Claim
Email	`email`	Email Claim
표시 이름	`name`	Name Claim
Realm 역할	`realm_access.roles`	Role Claim
전체 그룹 경로	`groups`	Group Claim

점으로 구분한 중첩 claim을 지원합니다. SSO 계정은 최초 로그인 때 생성되고 이후 로그인마다 프로필과 Keycloak 원천 역할을 재동기화합니다. 로컬 관리자가 추가한 역할은 `local`, IdP가 제공한 역할은 `keycloak` 원천으로 분리되어 IdP 역할 회수 시 로컬 예외 권한까지 우연히 삭제되지 않습니다.

사용자 화면에서는 로컬 계정 생성, 역할 관리, 비활성화, 잠금 해제, 비밀번호 초기화와 삭제를 수행합니다. 자기 잠금과 마지막 Super Admin 제거는 서버가 트랜잭션 안에서 차단합니다. 계정 비활성화·삭제는 Session과 API key를 함께 폐기하며, SSO 계정 비밀번호/프로필은 Keycloak에서 관리합니다. 긴급 접근을 위해 TOTP가 적용된 로컬 Super Admin 하나 이상을 Keycloak 장애 도메인 밖에 보관하십시오.

17. 운영 통계와 관리 콘솔 API 연결

운영 현황은 브라우저에서 임의 합산하지 않고 `GET /api/v1/dashboard/statistics`의 PostgreSQL 집계를 사용합니다. 이 API는 `assets.read` 권한이 필요하며 삭제되지 않은 자산만 집계합니다.

지표	판정 기준
자산 최신성	`last_seen_at`이 최근 24시간 이내
점검 필요 자산	전체 활성 자산 − 최근 24시간 확인 자산
정상 Agent	차단되지 않고 `last_seen_at`이 최근 30분 이내
점검 필요 Agent	전체 Agent − 정상 Agent
수집/실패	최근 24시간 `agent_events`와 `processing_status=failed`
7일 추이	UTC 날짜 기준 이벤트·실패 일별 합계, 빈 날짜도 0으로 반환

멀티 Pod에서도 모든 수치는 공용 PostgreSQL에서 계산되므로 sticky session이나 Pod 로컬 cache가 필요 없습니다. 운영 경보 기준은 조직의 수집 주기에 맞춰 별도 모니터링 시스템에서 정하되, 화면의 30분/24시간 기준은 일상 점검의 공통 baseline으로 사용하십시오.

관리 콘솔은 자산 CRUD·삭제/복원·이력·관계·병합/분리, Agent 등록·회전· 차단·mTLS·서명 업데이트, Query 검증/실행, Agent 자동 등록·PostgreSQL· Keycloak·일반 설정과 rollback, 사용자 수명주기, API key scope/회전/폐기, 감사 상세 API를 연결합니다. 권한이 없는 메뉴와 버튼은 숨기고 Server에서도 동일 RBAC를 다시 검증합니다. 로컬 로그인과 Keycloak callback은 동일한 SameSite CSRF cookie를 발급하며 콘솔 API client가 상태 변경 요청에 자동 적용합니다.

우측 상단 프로필 메뉴는 내 계정 보안, 개인화와 로그아웃을 연결합니다. 개인화는 사용자 ID별 브라우저 저장소에 테마, 정보 밀도, 로그인 시작 화면, 운영 통계 자동 갱신 주기와 모션 축소를 보관합니다. 이 값은 Server 정책이나 다른 브라우저에 전파되지 않으므로 보안·권한 설정 용도로 사용하지 않습니다.