Invenqor

관리자 가이드

수집 데이터 사전, 배포·인증·운영 통제와 장애 대응 기준서

버전	v0.2.33
작성일	2026-09-11
문서	관리자 가이드

> **대상 버전** Agent v0.2.33 · Server v0.2.33 · **문서 버전** 1.1 · **기준일** 2026-09-10 · **문서 등급** 공개

> Server v0.2.33 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 Primary입니다. PostgreSQL DSN을 지정한 배포는 연결 또는 migration에 실패하면 fail-closed로 기동·readiness가 실패하며 SQLite로 전환하지 않습니다. SQLite는 DSN을 지정하지 않은 단일 인스턴스 개발·복구 모드입니다.
- 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. 완료된 `processed` 이벤트는 다른 Pod의 늦은 실패 기록으로 `failed` 상태가 되지 않습니다.
- 원시 process·service·software.package 증거는 내장 카탈로그가 host별 `software_product` 로 자동 정규화합니다. 운영자는 수동 프로세스 매핑표 대신 제품의 설치·실행 상태, 확신도와 원천 근거를 관리합니다.
- 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 v0.2.33를 운영 환경에 배포하는 Linux·Windows, 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

v0.2.33에는 중앙 Server·대시보드·서명 자동 업데이트·자산 API·MCP와 주요 소프트웨어 자동 식별이 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

2.0 Windows 배포

Windows용 Agent는 `invenqor-agent-windows-x86_64.zip`으로 배포합니다. Linux판과 같은 수집 스키마·자산 키·등록 절차·서명 업데이트를 사용하므로 혼재 환경에서 Server 설정을 나눌 필요가 없습니다.

항목	값
대상	Windows 10/11, Windows Server 2016 이상, x86_64
설치 경로	`%ProgramFiles%\Invenqor\invenqor-agent.exe`
설정	`%ProgramData%\Invenqor\config.toml`
상태·큐	`%ProgramData%\Invenqor\state`
서비스	`invenqor-agent`, LocalSystem, 지연 자동 시작
네트워크	단일 HTTPS outbound. 수신 포트를 열지 않습니다
외부 의존	없음. 정적 링크 단일 실행 파일이며 .NET·VC++ 재배포 패키지가 필요하지 않습니다

로그 위치

Windows 서비스의 표준 오류는 어디에도 남지 않습니다. 그래서 Agent는 자신의 로그를 상태 디렉터리에 함께 씁니다.

```
Get-Content "$env:ProgramData\Invenqor\state\agent.log" -Tail 50 -Wait
```

8 MiB에서 `agent.log.1` 로 한 번 회전하므로 최대 16 MiB를 사용합니다. 서비스가 시작되지 않는 경우에는 Windows **시스템** 이벤트 로그의 Service Control Manager 항목도 함께 확인하십시오. Agent 프로세스가 뜨기 전에 실패한 경우는 그쪽에만 기록됩니다.

`--diagnose` 는 서비스 상태와 마지막 수집 시각을 함께 보고합니다. Server에 도달 가능하다는 것과 Agent가 실제로 동작한다는 것은 다른 문제이고, 전자만 확인하면 서비스가 죽어 있는 호스트도 정상으로 보입니다.

LocalSystem으로 실행하는 이유는 권한 확대가 아니라 수집 범위입니다. Service Control Manager 열람, 모든 로드된 사용자 하이브의 설치 소프트웨어, 네트워크 어댑터 구성은 하위 권한 계정에서 읽을 수 없습니다.

설치:

```
# 관리자 PowerShell, 압축을 푼 디렉터리에서
.\scripts\install.ps1
notepad "$env:ProgramData\Invenqor\config.toml"   # [server] url 설정
Restart-Service invenqor-agent
& "$env:ProgramFiles\Invenqor\invenqor-agent.exe" `
  --config "$env:ProgramData\Invenqor\config.toml" --diagnose
```

`install.ps1` 은 멱등입니다. 다시 실행하면 바이너리만 교체하고 설정과 미전송 큐는 유지하며 권한을 복구합니다. 두 디렉터리는 `icacls` 로 SYSTEM과 Administrators로 제한합니다. 상태 디렉터리에는 장비 자격 증명이 있고 설정 파일에는 등록 Token이 들어갈 수 있으므로 일반 사용자가 읽을 수 있으면 안 됩니다. `--diagnose` 는 이 조건을 점검해 일반 사용자가 읽을 수 있으면 실패로 보고합니다.

동일 호스트의 별도 인스턴스처럼 기본값이 아닌 Windows 서비스명이 필요하면 최초 설치와 이후 업그레이드에 같은 값을 지정합니다.

```
$serviceName = 'Invenqor Agent Finance'
.\scripts\install.ps1 -ServiceName $serviceName
Restart-Service -Name $serviceName
```

Installer는 서비스명을 별도 argv로 인용한 `--service-run --service-name "<name>"` 을 SCM `binPath` 에 등록하고, 검증된 이름을 설정 옆 `%ProgramData%\Invenqor\service-name` 에 보존합니다. Agent는 이 파일을 읽어 SCM handler 등록, `--diagnose` , 콘솔 `--update-now` 재시작에 같은 이름을 사용합니다. 기존 기본 설치의 `--service` 명

령줄과 marker가 없는 상태도 계속 `invenqor-agent` 로 동작합니다. marker를 수동 편집하지 말고 Installer를 다시 실행하십시오. 앞뒤 공백, slash·backslash·따옴표·제어 문자가 있는 이름은 command line injection 방지를 위해 설치와 Agent 양쪽에서 거부됩니다.

WMI는 사용하지 않습니다. 통상적인 방법이지만 COM 아파트먼트와 Winmgmt 서비스 의존이 생기고, 필요한 값은 모두 레지스트리·Win32·SCM에서 얻을 수 있습니다. 특히 `Win32_Product` 는 조회할 때마다 설치된 모든 패키지에 Windows Installer 정합성 검사를 유발하므로 사용하지 않고, 설치 소프트웨어는 Uninstall 레지스트리 키에서 읽습니다.

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/releases/download/v0.2.33/invenqor-agent-linux-x86_64.tar.gz
curl -fLO https://github.com/hkjang/invenqor/releases/download/v0.2.33/invenqor-agent-linux-x86_64.tar.gz.s
sha256sum -c invenqor-agent-linux-x86_64.tar.gz.sha256
```

검증 결과가 `OK` 가 아니면 배포를 중단합니다. SHA-256은 전송 오류와 변조 탐지에 사용합니다. GitHub Release의 배포 패키지 자체에는 별도 공급망 attestation이 포함되지 않으므로 고통제 환경에서는 승인된 내부 저장소로 반입한 뒤 조직 서명을 추가하십시오. 이는 중앙 Agent 자동 업데이트용 dual-signature v2 bundle과 다른 통제입니다. 자동 업데이트 bundle은 관리자가 오프라인 Ed25519 키로 배포할 실행 artifact를 직접 서명할 때 생성합니다.

폐쇄망에 세 플랫폼을 함께 반입할 때는 `invenqor-agents-0.2.33.tar.gz` 와 같은 이름의 `.sha256` 을 받으십시오. 이 묶음에는 Linux x86_64·aarch64, Windows x86_64 패키지와 각 체크섬, `sign-agent-update-manifest-v2.py` 가 들어 있습니다.

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	scheme·host·선택 port만 있는 Server origin. path/query/fragment/인증정보 금지
`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

외부 HTTPS Ingress가 기본 443을 제공하면 `server.url`에는 외부 origin만 쓰고 내부 Service port `7070`을 노출하지 않습니다. 자동 update 활성화 시 공개키는 유효한 Base64 Ed25519 32-byte key, 설치 경로는 절대 경로여야 하며 위반하면 Agent가 기동 단계에서 설정 오류로 종료합니다.

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.

4.5 등록·연동 상태 확인 수단

등록 실패는 Server에 아무 기록도 남지 않을 수 있으므로 Agent 측에 항상 남는 경로를 함께 제공합니다.

수단	상태 변경	종료 코드	용도
`--diagnose`	없음	실패 항목이 있으면 1	설정·상태 디렉터리·DNS·도달성·Server 정책·자격 증명을 순서대로 판정
`--status`	없음	등록·전송이 밀려 있으면 1	마지막 실패 코드, Server `request_id`, 큐 깊이 확인
`status.json`	매 주기 기록	—	저널 열람이 불가능한 환경의 사후 분석
`--once`	수집·전송 1회	전송 실패 시 2	설치 자동화의 검증 단계
`--check-update`	스테이징만	확인 실패 시 1	서명 검증까지만 수행하고 설치하지 않음
`--update-now`	설치까지 1회	권한 부족 시 3	한 대를 즉시 갱신하거나 canary를 손으로 확인
`--apply-pending-update`	설치	실패 시 1	스테이징된 업데이트를 권한 있는 계정으로 적용

`--diagnose`와 `--status`는 등록도, 전송도, 자격 증명 교체도 하지 않으므로 운영 중 아무 때나 실행할 수 있습니다. `--once`는 운영 서비스와 같은 상태 디렉터리를 동시에 사용하므로 서비스를 멈춘 뒤 실행합니다.

`status.json` 주요 필드:

필드	의미
`enrollment.state`	`not_configured`, `pending`, `enrolled`, `failed`
`enrollment.last_error`	코드, Server `request_id`, 조치 문구
`delivery.consecutive_failures`	연속 전송 실패 횟수
`queue.pending_events`, `queue.bytes`	미전송 이벤트 수와 크기
`collection.collector_errors`	마지막 주기의 수집기 실패 수
`updates.enabled`, `updates.channel`	자동 업데이트 설정 상태
`updates.running_version`, `updates.staged_version`	실행 중 버전과 재시작 시 적용될 버전
`updates.last_error`	마지막 업데이트 실패 코드와 조치 문구

감시 시스템에서는 다음 한 줄로 판정할 수 있습니다.

```
/opt/invenqor-agent/bin/invenqor-agent \
  --config /etc/invenqor-agent/config.toml --status --json |
  jq -e '.enrollment.state == "enrolled" and (.delivery.last_error == null)'
```

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 v0.2.33 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
`ifindex`	커널 인터페이스 인덱스
`addresses`	IPv4/IPv6 주소 문자열 배열

네트워크 네임스페이스 기준으로 보이는 인터페이스만 수집합니다. 프리픽스 길이, 브로드캐스트, VLAN/본딩 관계는 v0.2.33에 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 v0.2.33의 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.

6.12 Windows 수집 차이

Windows에서도 같은 카테고리를 만들되, 원천과 몇몇 필드가 다릅니다.

카테고리	Windows 원천	Linux와 다른 필드
`system`	`CurrentVersion` 레지스트리, SMBIOS	`os_name`, Linux 호환 `os_release`, `os_build` (UBR 포함), `edition_id`, `install_type`, `domain_role`, `firmware_uuid`
`hardware.cpu`	`CentralProcessor\0` 레지스트리, `GetNativeSystemInfo`	`megahertz`
`hardware.memory`	`GlobalMemoryStatusEx`	`page_file_total_bytes`, `page_file_available_bytes`
`hardware.filesystem`	`GetLogicalDriveStringsW`, `GetVolumeInformationW`	`mountpoint` 은 `C:\` 형태, `drive_type`, `label`
`network.interface`	`GetAdaptersAddresses`	`interface_type`, `dns_suffix`, `operational`
`process`	`CreateToolhelp32Snapshot`	`start_ticks` 는 생성 FILETIME
`software.package`	Uninstall 레지스트리 3개 범위	`publisher`, `scope`, `installer`, `install_location`
`service`	Service Control Manager + 서비스 레지스트리 키	`start_type`, `run_as`, `delayed_auto_start`, `image_path`
`account.user`	`NetUserEnum` 레벨 3, `NetLocalGroupGetMembers`	`uid` 는 RID, `groups`, `administrator`, `disabled`, `locked_out`
`container.environment`	런타임 서비스 등록	`runtimes`, `windows_containers_feature`

판단 근거가 되는 세 가지를 특히 확인하십시오.

- **운영체제 이름**: 모든 Windows 11 호스트는 레지스트리 `ProductName` 에 "Windows 10 …"을 보고합니다. Microsoft가 값을 갱신하지 않았기 때문이며, 이 값을 그대로 쓰는 인벤토리는 Windows 11 전체를 Windows 10으로 집계합니다. Agent는 빌드 번호 22000 이상에서 이름을 교정합니다. Server 계열은 빌드 번호 체계를 공유하므로 교정 대상이 아닙니다. v0.2.33 Agent는 최상위 `os_name` 과 `os_release.{id,name,pretty_name,version_id,build_id}` 를 함께 보내고, Server는 두 형식을 모두 읽습니다. v0.2.13 이하에서 이미 등록된 장비는 저장된 `system` 원천을 첫 heartbeat에서 다시 투영하거나 다음 `system` delta를 처리해 자동 복구하므로 Agent 상태 파일을 삭제하지 마십시오.
- **설치 소프트웨어 범위**: 64비트 machine 뷰만 읽으면 32비트 제품(WOW6432Node) 과 사용자별 설치가 모두 누락됩니다. 세 범위를 모두 읽고 `scope` 필드로 구분합니다. Windows가 프로그램 목록에서 숨기는 항목 — `SystemComponent`, 업데이트·핫픽스 `ReleaseType`, 번들 설치 프로그램의 자식 항목 — 은 제외합니다. 제외하지 않으면 패치된 서버에서 수천 건의 핫픽스가 실제 소프트웨어를 덮습니다.
- **서비스**: 현재 상태(`state`)와 시작 유형(`start_type`)을 함께 보고합니다. 멈춰 있는 서비스가 멈춰 있어야 하는 것인지는 상태만으로 답할 수 없습니다.

프로세스 명령행은 Windows에서도 기본 비수집입니다. 서비스와 예약 작업의 명령행에 자격 증명이 들어가는 경우가 흔합니다.

6.13 주요 소프트웨어 자동 식별 (`software_product`)

원시 `process` 는 시점 관찰이고 `service` 와 `software.package` 는 설치·구성 증거입니다. 어느 하나만 CMDB 소프트웨어로 사용하면 다중 PID를 중복 집계하거나, 설치됐지만 실행되지 않는 제품과 실행됐지만 패키지 관리자를 거치지 않은 제품을 놓칩니다. v0.2.33 Server는 매 inventory에서 세 원천을 host 단위로 다시 결합해 `software_product` 대표 자산을 만듭니다.

출력 필드	의미
`product_key` , `product_name` , `vendor` , `role`	카탈로그의 안정 ID, 표시명, 제조사와 운영 역할
`version` , `versions`	패키지 증거에서 확인된 대표/전체 버전
`install_state`	패키지·서비스가 있으면 `installed` , 프로세스만 있으면 `observed`
`runtime_state`	`running` , `stopped` , 현재 상태를 확인할 수 없는 `unknown`
`service_names` , `process_names` , `package_names`	정규화에 사용한 host별 신호
`executable_paths`	인자를 제거한 실행 파일 경로. 서비스 비밀번호·Token을 복제하지 않음
`evidence[]`	`kind` , 이름, 원시 `source_asset_id` 로 구성된 설명 가능한 근거
`confidence` , `detection_method` , `catalog_version`	근거 강도, `builtin_catalog` , 규칙 집합 버전

내장 카탈로그 `2026.08.1` 은 51개 제품을 보수적으로 식별합니다. DB, 검색, 웹·proxy, 애플리케이션 서버, 컨테이너·오케스트레이션, 메시지 브로커, 관측·보안, 백업, CI/CD, 원격 접속, guest tools와 Invenqor Agent에 더해 Office/Microsoft 365, Chrome·Edge·Firefox, Teams·Zoom, Java·.NET, MECM·Tanium·BigFix, Elastic Agent·Wazuh를 포함합니다. 카탈로그에 없는 프로세스는 제품으로 만들지 않으므로 운영자가 일반 PID를 수동 분류할 필요가 없고, 알 수 없는 이름 때문에 오탐 자산이 늘어나지 않습니다. Chrome·Java 같은 범용 process는 단독 신호로 제품에 승격하지 않고 패키지·서비스·고유 경로 등 강한 증거를 요구합니다.

판별 기본 강도는 서비스명 0.95, 패키지명 0.90, 실행 파일 basename 0.86, 제품 고유 경로 0.84, 프로세스명 0.82입니다. 서로 다른 증거 종류가 일치하면 가산하되 0.99를 넘지 않습니다. PostgreSQL client/common/libs/devel처럼 서버 설치를 뜻하지 않는 패키지는 제외합니다. 일반 부분 문자열이 아니라 정확 이름과 제한된 glob, 실행 경로를 사용합니다. 화면에서 0.80 미만은 **검토 권장**으로 분리하지만 자동 제거하거나 사용자가 매핑을 보정하도록 강제하지 않습니다.

제품 자산은 다음 운영 계약을 따릅니다.

1. Agent ID와 `product_key`로 host별 안정 자산 키를 만들고 여러 PID를 하나로 합칩니다.
2. 실제 host와 자동 `runs_on` 관계를 만들고 host 환경을 상속합니다.
3. Agent inventory와 같은 DB 트랜잭션에서 갱신하므로 멀티 Pod에서도 원시 증거와 정규화 상태가 어긋난 중간 결과를 노출하지 않습니다.
4. 증거가 바뀌면 상태·버전·확신도를 다시 계산하고 변경 이력 사유 `automatic_software_catalog`를 기록합니다.
5. 마지막 증거가 사라지면 제품과 자동 관계를 논리 종료하고, 다시 나타나면 기존 제품을 재활성화합니다. 같은 inventory 재처리는 중복을 만들지 않습니다.

일반 자산 화면과 운영 Dashboard는 `scope=managed`를 사용해 process 자산을 기본 제외합니다. 이는 표시·집계 범위일 뿐 원시 evidence를 삭제하지 않습니다. 사고 조사에는 **프로세스 관찰 포함**을 켜고, 일상 운영에는 **주요 소프트웨어** 화면에서 제품·호스트·버전·설치/실행 상태와 상세 근거를 사용하십시오.

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/0.2.33
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
`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": "2026-08-24.1"
}
```

HTTP 2xx와 `accepted: true`가 모두 충족돼야 성공입니다. `policy_version`은 로그에 관찰만 하며 v0.2.33는 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

- GitHub 배포 아카이브는 SHA-256을 제공하지만 별도 vendor package 서명, SBOM, provenance는 없음. 운영자가 생성하는 Agent 자동 업데이트 v2 서명과는 별도 통제
- 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

11.4 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 API·Agent 요청의 method, path, status, 응답 크기, 처리 시간, source IP와 request ID를 구조화 access log로 저장합니다. 정상 liveness/readiness probe와 정적 UI asset은 조사 가치를 유지하면서 보존 한도를 소모하지 않도록 제외하지만, 실패 응답은 경로와 관계없이 기록합니다. Agent 등록, 정책 거부, 인증·schema·처리 실패와 Server 내부 오류도 같은 화면에 합쳐집니다. Token, Secret, Authorization, URL password는 기록 전에 redaction하고 원문 인벤토리는 복제하지 않습니다. 기본 보존은 30일 또는 최신 10,000건 중 먼저 도달하는 한도입니다. DB 연결 전 시작 실패와 프로세스 종료 같은 컨테이너 lifecycle stdout은 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.

11.5 멀티 Pod Ingest 멱등성

Agent Event의 상태 판정은 다음과 같습니다.

상태	의미	운영 조치
`processing`	한 Pod가 처리권을 보유	짧게 기다린 뒤 동일 Event ID로 재전송
`pending`, `failed`	재처리 가능	`processing_error` 와 request ID를 조사한 뒤 재전송
`processed`	성공한 최종 상태	재전송해도 중복 성공이며 수동 상태 변경 금지

v0.2.33 Server는 실패 상태를 기록하는 UPSERT에서 기존 `processed` 행을 제외합니다. 따라서 Pod A가 성공을 commit한 직후 Pod B가 같은 이벤트 처리 실패를 기록해도 상태와 자산 projection은 성공 상태로 유지됩니다. 한 요청이 오류를 반환했더라도 DB가 `processed` 이면 Agent의 at-least-once 재전송에 맡기고, 행을 `failed` 로 되돌리거나 동일 payload를 새 Event ID로 복제하지 마십시오.

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

v0.2.33는 관리자가 승인한 Ed25519 manifest v2 서명 artifact의 자동 스테이징, Windows service 자동 적용과 Linux 권한 분리 기반 원자 교체를 지원합니다. 서명 개인키는 Server와 Agent에 배포하지 않고 오프라인 환경에서 보관합니다. v2 서명은 버전·channel·OS·architecture·크기·SHA-256· `allow_downgrade` 를 함께 보호합니다. Agent는 이 계약이 모두 맞을 때만 `pending.json` 을 생성합니다.

설치 직전에 Agent는 스테이징된 바이너리를 최대 10초 동안 실행하고 stdout/stderr를 각각 64 KiB로 제한해 `--version` 이 약속한 정확한 출력을 확인합니다. 이 범위를 넘기거나 출력이 다르면 설치를 중단하고 기존 바이너리를 유지하므로, 실행되지 않는 빌드가 fleet의 수집을 멈추지 않습니다. 교체된 기존 바이너리는 `.previous` 로 보존됩니다.

적용 시점은 init 시스템에 따라 다릅니다. systemd는 감시 경로로 즉시, OpenRC와 SysV는 서비스 시작 시 적용합니다.

1. 새 바이너리와 SHA-256, 변경 내역, 설정 호환성을 승인합니다.
2. 오프라인 Ed25519 키로 canonical manifest v2를 서명합니다. helper가 artifact의 정확한 크기와 SHA-256을 계산하므로 수동으로 메시지를 조립하지 마십시오.

   ```
   python3 scripts/sign-agent-update-manifest-v2.py \
     --artifact invenqor-agent-linux-x86_64 \
     --private-key update-signing.pem \
     --version 0.2.33 --channel stable \
     --os linux --architecture x86_64 \
     > invenqor-agent-linux-x86_64.signature-bundle.json
   ```

   출력 JSON에는 이전 Agent의 정상 상향 업데이트용 artifact 서명과 metadata-bound v2 manifest 서명이 모두 들어 있습니다. 레거시 자동화가 두 raw 파일을 요구할 때만 `--signature-output` 과 `--manifest-signature-output` 을 선택적으로 지정하십시오.

   Server에 `INVENQOR_UPDATE_PUBLIC_KEY` 가 없으면 게시 API는 `UPDATE_SIGNING_KEY_MISSING` 으로 fail-closed 거부되고 콘솔 게시 버튼도 잠깁니다. 공개키가 있더라도 두 서명 중 하나가 맞지 않으면 `UPDATE_SIGNATURE_REJECTED` 로 거부됩니다.
3. 콘솔 **운영 → Agent 업데이트**에서 서명과 동일한 버전·channel·OS·architecture· rollback 여부와 rollout 비율(초기 10%)로 게시합니다. Artifact와 helper가 만든 `.signature-bundle.json` 하나를 올리면 서명된 식별 필드는 자동으로 채워지고 잠깁니다. 이 JSON을 승인 기록에도 보관하십시오.

   Windows판은 `os` 를 `windows` 로 지정해 별도로 게시합니다. 릴리즈는 버전·OS· 아키텍처로 구분되므로 같은 버전의 Windows판과 Linux판이 공존하고, 각 Agent는 자신의 플랫폼 릴리즈만 받습니다. 서명은 플랫폼별 실행 파일 각각에 대해 따로 만들어야 합니다.
4. canary에서 설치·수집·전송·재시작을 확인합니다. 특정 한 대를 즉시 확인하려면 해당 호스트에서 `--update-now` 를 실행합니다.
5. 콘솔의 적용 대수와 fleet 버전 분포, 중앙 오류율을 확인한 뒤 25 → 50 → 100%로 확산합니다. 재업로드는 필요하지 않습니다.
6. 문제가 발견되면 **중단**(rollout 0)을 누릅니다. 이미 받은 호스트는 12.3에 따라 되돌립니다.

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

Windows에서의 적용 경로

Windows는 실행 중인 실행 파일을 덮어쓰거나 삭제할 수 없지만 **이름을 바꿀 수는 있습니다**. Agent는 먼저 스테이징된 candidate를 제한된 환경에서 실행해 약속한 버전을 보고하는지 확인합니다. 자기 점검이 성공한 뒤에만 기존 바이너리를 `invenqor-agent.exe.previous`로 옮기고 candidate를 원래 경로에 원자 배치합니다. 실패하면 live 실행 파일은 건드리지 않고 candidate만 폐기합니다.

교체 후에는 재시작이 필요합니다. 서비스는 스스로를 중지하고 시작할 수 없으므로 (SCM은 중지 중인 서비스의 시작을 처리하지 않습니다) `SERVICE_STOPPED`를 정상 보고하지 않고 종료합니다. `install.ps1`이 설정한 SCM crash recovery가 fresh install 직후에도 새 바이너리로 즉시 되살립니다.

v0.2.33 service의 background checker는 서명된 update를 내려받은 뒤 이 적용 경로를 자동 실행합니다. 따라서 운영자가 각 Windows 장비에서 `--update-now`를 호출하거나 재부팅할 필요가 없습니다. `Content-Length`가 없는 chunked Ingress artifact도 manifest 크기 이하로 bounded streaming하고 실제 길이·SHA-256·서명을 재검증합니다.

```
sc.exe qfailure invenqor-agent     # 복구 동작 확인
Get-Service invenqor-agent ¦ Select-Object Status, StartType
```

콘솔에서 `--update-now`를 실행하면 교체 후 SCM으로 서비스를 직접 재시작합니다. 바이러스 검사기가 새로 쓴 파일을 잠시 잠그는 일이 흔하므로 이름 변경은 몇 초간 재시도합니다. 릴리즈는 `os`와 아키텍처로 구분되므로 같은 버전의 Windows판과 Linux판이 공존하고, Agent는 자신의 플랫폼 릴리즈만 받습니다.

12.3 롤백

장비 수가 적으면 호스트에서 직접 되돌립니다. 여러 대를 되돌려야 하면 이전 버전을 `allow_downgrade`로 게시합니다. helper에도 `--allow-downgrade`를 지정해 그 의도를 v2 서명에 포함해야 합니다. Agent는 이 표시가 서명된 릴리즈만 하위 버전으로 받아들입니다. 기존 artifact-only(v1) 서명은 정상 상향 업데이트만 허용되며 롤백 권한으로 사용할 수 없습니다.

1. 서비스 중지
2. 이전 검증 바이너리 복원(`<바이너리>.previous`)
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. `--validate-config` 를 **서비스 계정으로** (`sudo -u invenqor-agent`) 실행하는 것이 중요합니다. root로 실행하면 서비스가 읽지 못하는 파일도 통과합니다.

13.1.1 설정 파일을 읽지 못하는 경우

`server.url` 이 설정되어 있는데도 로그에 `no configuration file was found` 가 남고 큐만 늘어난다면, 파일은 있으나 서비스 계정이 읽지 못하는 상태입니다. 디렉터리에 실행(traverse) 권한이 없으면 파일 권한이 맞아도 읽을 수 없습니다.

```
sudo namei -l /etc/invenqor-agent/config.toml
sudo chown root:invenqor-agent /etc/invenqor-agent /etc/invenqor-agent/config.toml
sudo chmod 0750 /etc/invenqor-agent
sudo chmod 0640 /etc/invenqor-agent/config.toml
sudo systemctl restart invenqor-agent
```

v0.2.13 이후 Agent는 이 상태에서 기본값으로 넘어가지 않고 조치 명령과 함께 기동을 거부하며, `--diagnose` 는 서비스 계정이 읽을 수 있는지까지 판정합니다.

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

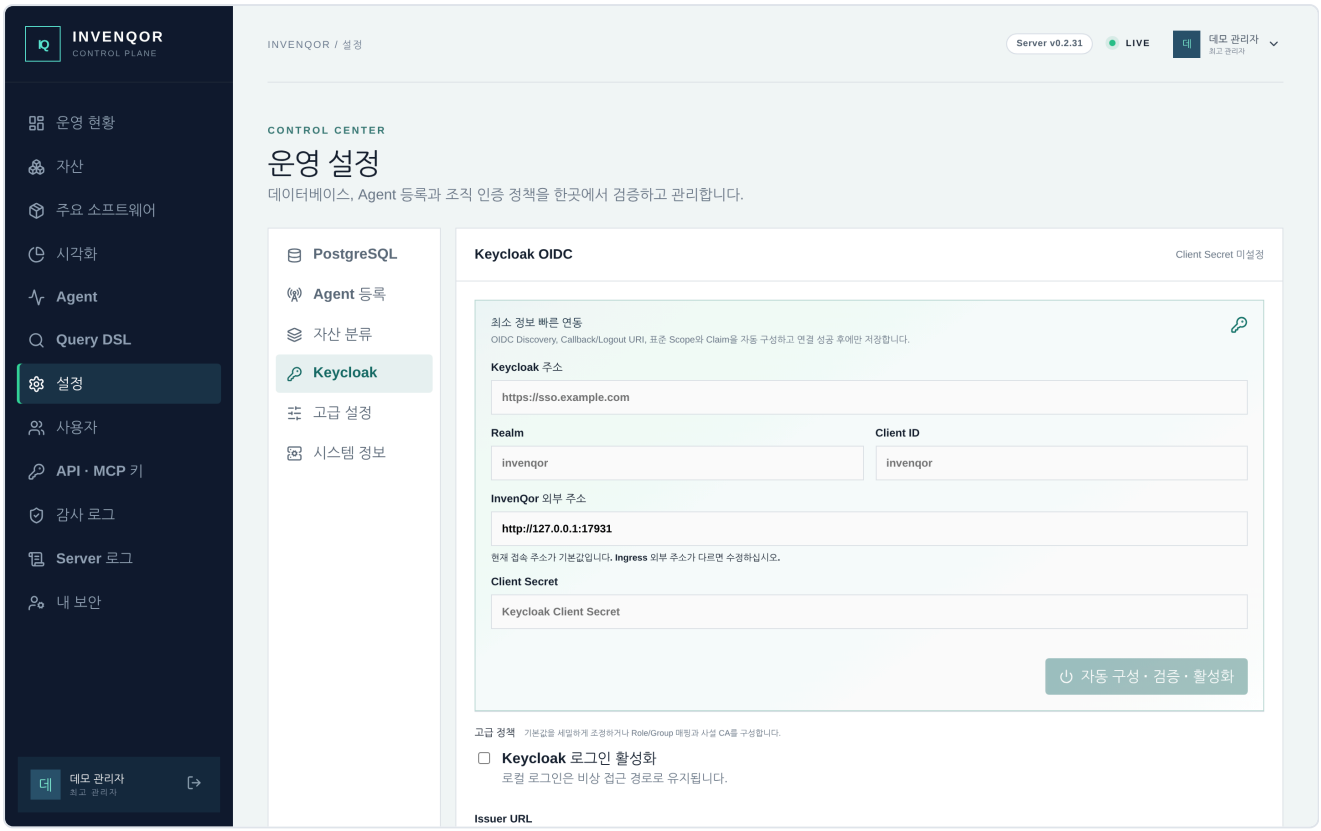
Server 관리 콘솔의 **설정 → Keycloak**은 OIDC Issuer/Realm, confidential client, Redirect URI, Scope, claim, Email domain, Role/Group mapping과 사설 CA를 관리합니다. 연결 테스트는 실제 discovery와 TLS 신뢰를 확인하며 Client Secret은 Master Key로 암호화되어 구성 여부만 표시됩니다. Client Secret 없이 SSO를 활성화하거나 존재하지 않는 내부 역할을 mapping하는 설정은 거부됩니다.

v0.2.33부터 Client Secret은 Pod 로컬 파일이 아니라 공용 PostgreSQL의 `auth.keycloak.client_secret`에 ciphertext envelope로 저장됩니다. 모든 Pod가 공유하는 32-byte Master Key를 사용한 AES-256-GCM AEAD이며 암호화 용도도 associated data로 인증합니다. 따라서 어느 Pod가 로그인 시작·callback을 처리해도 같은 Secret을 사용합니다. Master Key가 다르면 Server는 임의로 새 Secret을 만들거나 평문으로 대체하지 않고 시작 또는 복호화 오류를 명시적으로 반환합니다.

v0.2.14의 Pod-local `bootstrap.enc` 만 있는 경우, 동일 Master Key와 기존 state PVC를 가진 첫 v0.2.33 Pod가 시작하면서 공용 DB로 자동 이관합니다. 두 번째 Pod의 Keycloak 설정에서 **Client Secret 구성됨**을 확인할 때까지 기존 PVC를 폐기하지 마십시오. PVC가 먼저 유실되어 공용 Secret도 없으면 로컬 Super Admin으로 전용 Keycloak 화면에서 다시 입력합니다. Master Key가 유실된 경우에는 일치하는 DB·Key 백업 복원이 우선이며, 상세 절차는 Server 설치 및 운영 가이드를 따릅니다.

일반 설정 목록·이력은 `auth.keycloak`과 `auth.keycloak.client_secret`을 노출하지 않습니다. `/api/v1/admin/settings`를 통한 두 키의 PATCH·rollback은 `409 DEDICATED_SETTING_ENDPOINT`로 거부되므로 자동화도 `/api/v1/admin/settings/keycloak` 전용 API를 사용해야 합니다. DB 행을 직접 수정하거나 마스킹 문자열을 Secret으로 다시 저장하지 마십시오.

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.

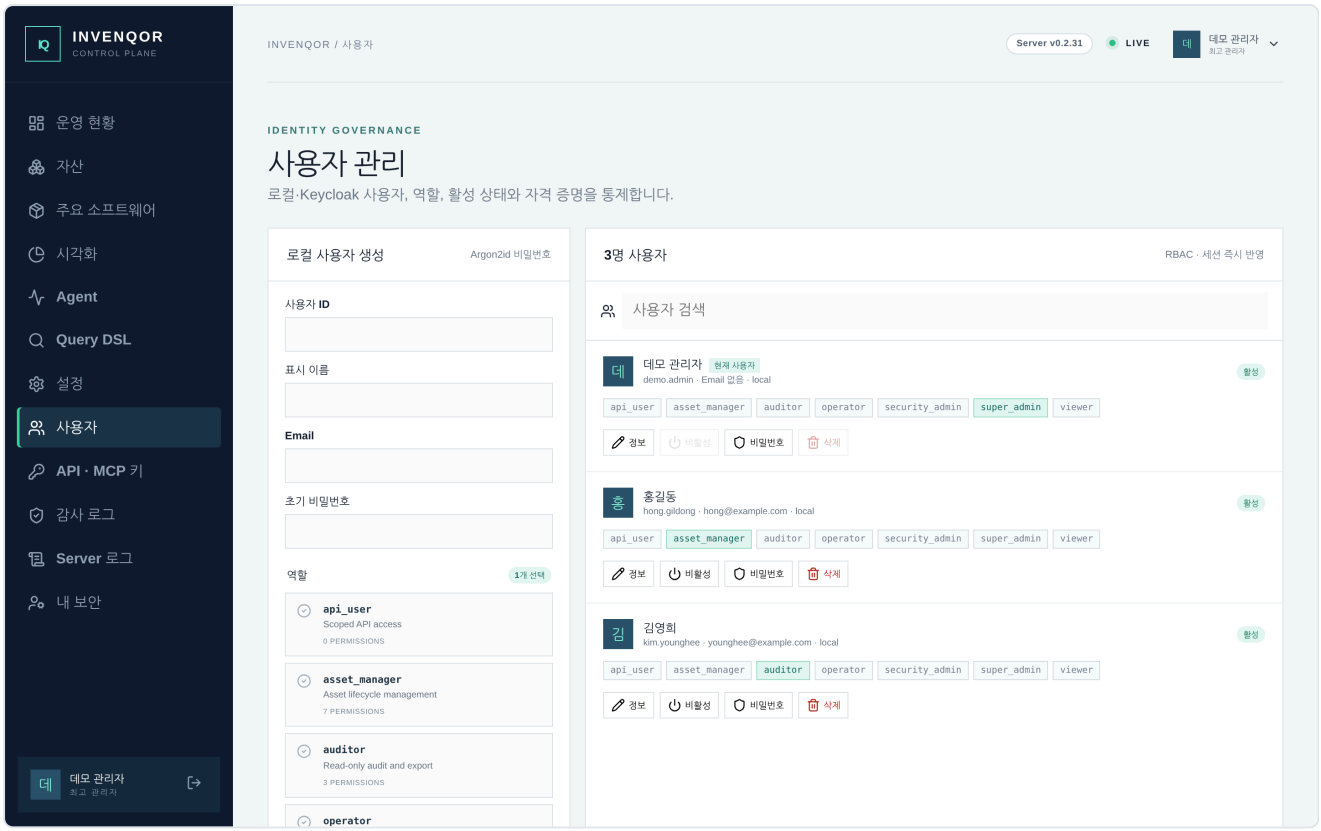


설정 → Keycloak — 최소 정보 빠른 연동으로 Keycloak 주소·Realm·Client 정보만 입력하고 Discovery 검증 뒤 활성화한다

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 관리 — 로컬 계정을 만들고 역할을 부여하며, 계정별로 정보 수정·비활성·비밀번호 초기화·삭제를 수행한다

내장 역할과 각 역할이 받는 권한은 다음과 같습니다. 역할은 시스템 역할이며 콘솔에서 권한 구성을 변경할 수 없습니다.

역할	설명	부여되는 권한
`super_admin`	모든 권한	정의된 모든 권한(`api_keys.manage` , `mcp.access` 포함)
`asset_manager`	자산 수명주기 관리	`assets.read` `assets.write` `assets.delete` `assets.merge` `assets.export` `relations.read` `relations.write`
`operator`	Agent·수집 운영	`agents.read` `agents.manage` `assets.read`
`security_admin`	보안 자산 검토	`assets.read` `agents.read`
`auditor`	읽기 전용 감사·내보내기	`assets.read` `assets.export` `audit.read`
`viewer`	읽기 전용 자산 조회	`assets.read`
`api_user`	Scope 기반 API 접근	없음(권한은 API key의 scope에서 나옵니다)

`api_user` 는 콘솔 권한을 부여하지 않습니다. 이 역할을 받은 계정이 할 수 있는 일은 그 계정이 소유한 API key의 scope로만 결정되므로, 연계 시스템 전용 계정에 사용하십시오.

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.

역할과 계정 이름 규칙:

- 권한은 역할 부여에서만 나옵니다. 마지막 역할을 회수하는 변경은 `ROLE_REQUIRED` 로 거부됩니다. 접근을 차단할 때는 계정을 비활성화하십시오. Keycloak이 부여한 역할이 남아 있으면 로컬 역할은 모두 회수할 수 있습니다.
- 계정 삭제는 논리 삭제이지만 사용자 ID는 즉시 해제되어 재사용할 수 있습니다. 삭제된 행은 `원래이름#deleted-〈id〉` 로 보존되어 감사 추적과 외부 참조가 유지됩니다.
- Keycloak 사용자명이 기존 로컬 계정과 겹치면 자동 연결하지 않고 `KEYCLOAK_USERNAME_CONFLICT` 로 거부합니다. 이름만으로 자동 연결하면 디렉터리 쪽 이름 변경으로 로컬 계정을 탈취할 수 있기 때문입니다. 로컬 계정을 정리하거나 다른 username claim을 사용하십시오.

16.1 Session Cookie와 SSO 왕복

Session과 CSRF Cookie는 `SameSite=Lax` 로 발급됩니다. Keycloak Callback은 외부 사이트에서 되돌아오는 이동이므로 `Strict` 에서는 Cookie가 전송되지 않아 SSO 직후 콘솔이 로그아웃 상태로 보입니다. 상태를 바꾸는 모든 요청은 여전히 `X-CSRF-Token` 헤더와 Cookie 이중 확인을 요구합니다.

`Secure` 속성은 요청이 HTTPS일 때만 부여합니다. 브라우저는 평문 HTTP로 전달된 `Secure` Cookie를 저장하지 않으므로, 폐쇄망 HTTP 설치에서 로그인이 성공한 듯 보이고도 실패하는 문제를 방지하기 위한 동작입니다. TLS를 상위 Proxy에서 종료할 때는 `X-Forwarded-Proto: https` 를 전달하도록 구성하십시오.

16.2 API Key scope·회전 충돌 처리

API Key 변경은 Pod 메모리 lock이 아니라 PostgreSQL의 현재 값을 조건으로 처리합니다. Load Balancer가 요청을 서로 다른 Pod로 보내도 다음 계약이 유지됩니다.

작업	동시성 보호	`409 API_KEY_CONFLICT` 이후 조치
Scope 추가·삭제	`scopes_json` CAS, 최대 8회 내부 재시도	최신 Key를 읽고 의도한 차이만 다시 적용
Scope 전체 교체	읽은 `scopes_json` 과 일치할 때 단일 UPDATE	최신 전체 목록에서 목표 scope를 재계산
이름+scope PATCH	이름과 scope를 같은 조건부 UPDATE로 반영	이름도 미반영이므로 최신 상태에서 요청 재작성
Secret 회전	현재 `key_hash` CAS	승자 Secret 배포 여부를 확인한 뒤 필요할 때만 새 회전

회전 충돌 응답에는 새 Secret이 없습니다. 따라서 `409` 응답에서 Secret을 추출하거나 그 값을 배포해서는 안 됩니다. 단순 자동 재시도는 이미 승리한 회전을 다시 회전시킬 수 있으므로, Key의 prefix·변경 시각과 작업 기록을 먼저 대조합니다. Scope 변경은 bounded exponential backoff와 jitter를 사용할 수 있지만 전체 목록을 이전 응답에서 그대로

재전송하면 다른 관리자의 변경을 덮어쓸 의도가 될 수 있으므로 항상 최신 상태에서 다시 계산하십시오.

17. 운영 통계와 관리 콘솔 API 연결

운영 현황은 브라우저에서 임의 합산하지 않고 `GET /api/v1/dashboard/statistics?scope=managed` 의 PostgreSQL 집계를 사용합니다. 이 API는 `assets.read` 권한이 필요하며 삭제되지 않은 자산 중 원시 process를 제외합니다. 전체 관찰 자산 집계가 필요한 외부 연계는 `scope` 를 생략하십시오.

주요 소프트웨어는 `GET /api/v1/assets/software-products` 를 사용합니다. `q` , `role` , `vendor` , `runtime_state` , `confidence` , `limit` , `offset` 필터를 지원하고 제품·인스턴스· 호스트·실행 상태·식별 품질 summary와 host별 evidence를 같은 응답에 제공합니다. 카탈로그 결과는 공용 PostgreSQL에 저장되므로 어느 Pod에서 조회해도 동일합니다.

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

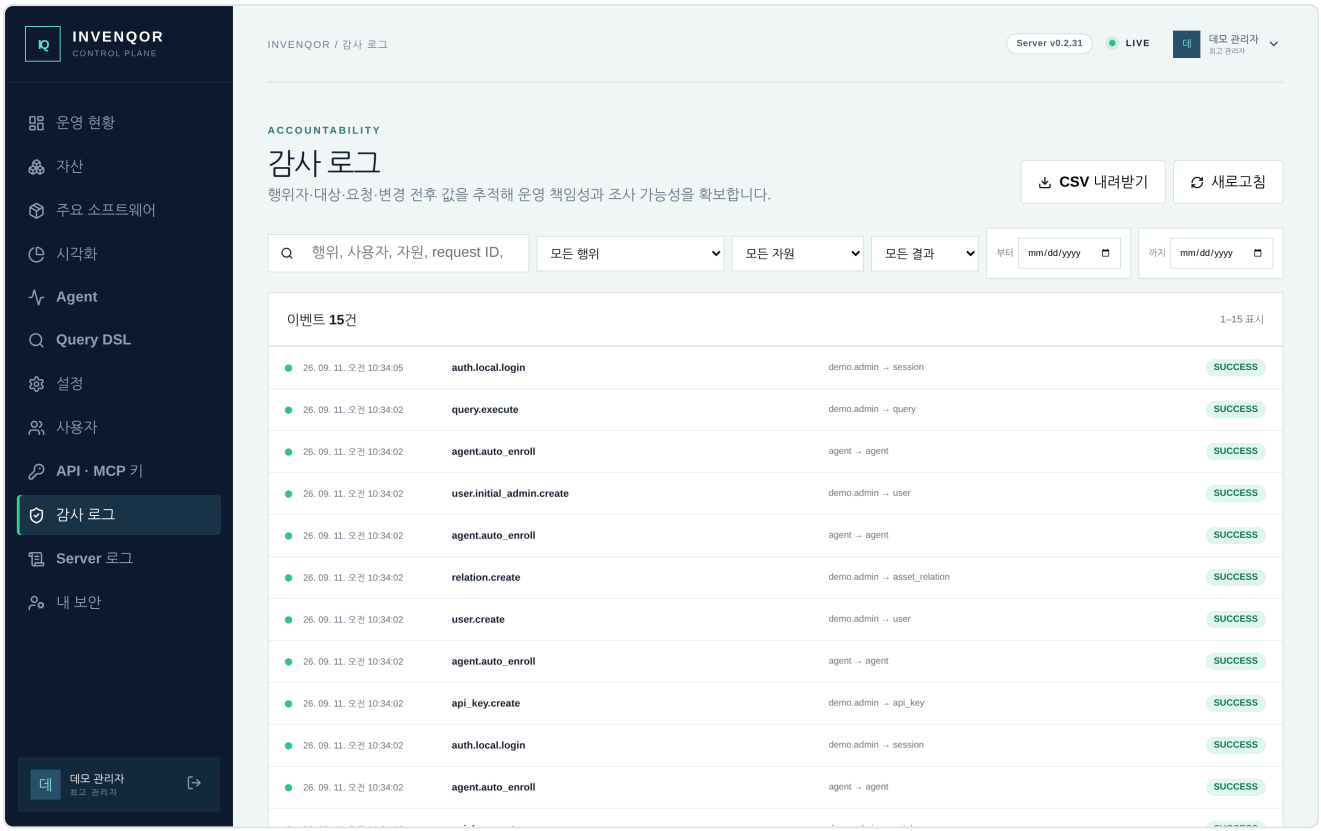
멀티 Pod에서도 모든 수치는 공용 PostgreSQL에서 계산되므로 sticky session이나 Pod 로컬 cache가 필요 없습니다. 운영 경보 기준은 조직의 수집 주기에 맞춰 별도 모니터링 시스템에서 정하되, 화면의 30분/24시간 기준은 일상 점검의 공통 baseline으로 사용하십시오.

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 화면 테마, 정보 밀도, 로그인 시작 화면, 운영 통계 새로고침 주기와 화면 움직임 최소화를 보관합니다. 이 값은 Server 정책이나 다른 브라우저에 전파되지 않으므로 보안·권한 설정 용도로 사용하지 않습니다.

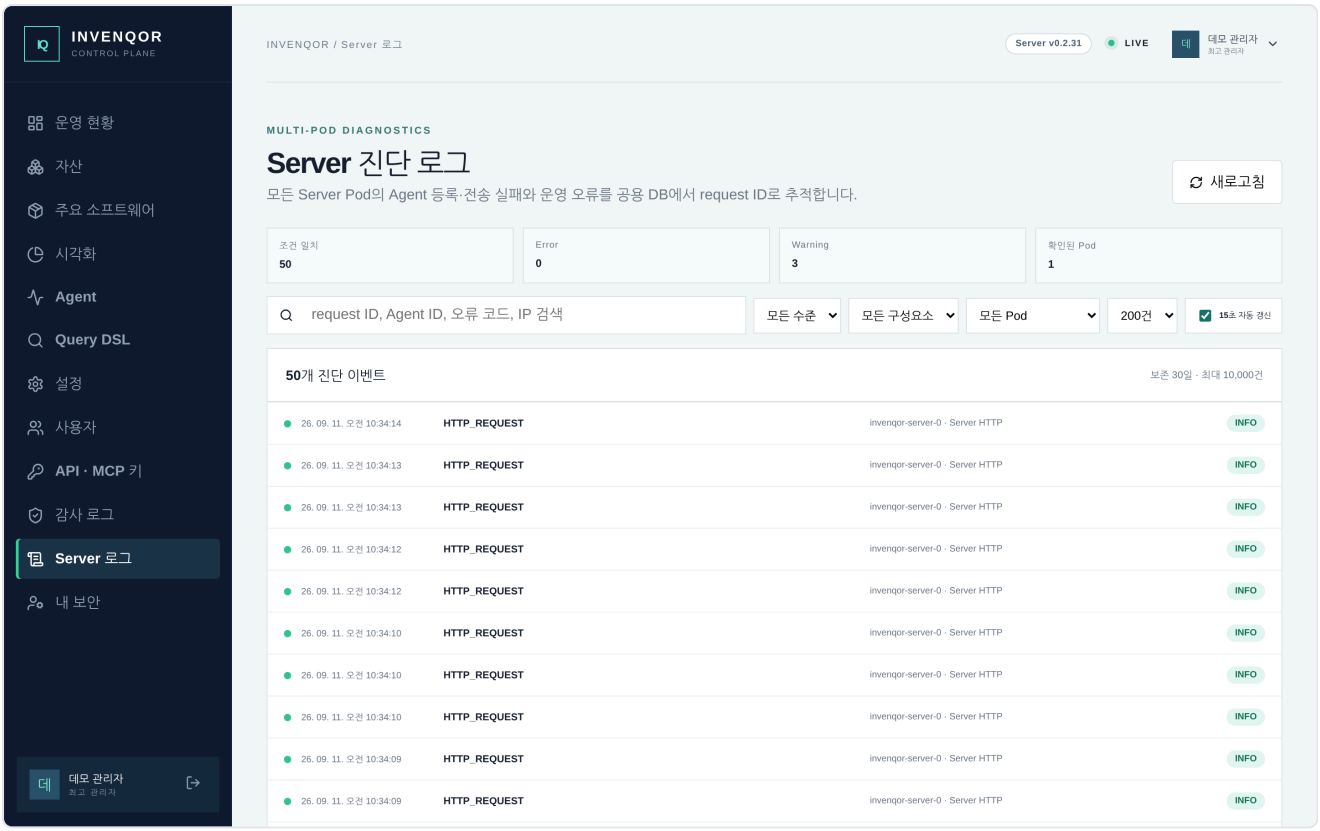
17.1 감사 로그와 Server 로그

감사 로그는 누가 무엇을 바꿨는지를 남깁니다. 행위·자원·결과·기간과 request ID로 검색하고, 총 건수와 페이징, CSV 내려받기를 제공합니다. Agent 자동 등록처럼 사람이 아닌 행위자도 같은 표에 기록됩니다.



감사 로그 — 행위·자원·결과·기간으로 검색하고 행마다 행위자와 대상, 결과를 확인한다

Server 로그는 모든 Pod의 Agent 등록·전송 실패와 Server 오류를 같은 공용 DB에서 검색합니다. 감사 로그 행의 request ID로 이동하면 그 요청이 남긴 진단 기록을 바로 볼 수 있습니다. 보존 일수와 최대 건수는 화면 오른쪽 위에 표시됩니다.



Server 로그 — 구성요소·Pod·수준으로 진단 이벤트를 검색하고 request ID로 요청을 추적한다

18. Server 기동 환경 변수

Server 프로세스가 읽는 설정은 아래가 전부입니다. 이 표는 `server/internal/config/config.go` 의 `Load()` 에서 그대로 옮긴 것으로, 여기에 없는 이름은 Server가 읽지 않습니다. 운영 중 바뀌는 값(Agent 등록 정책, Keycloak, 분류 규칙 등)은 환경 변수가 아니라 관리 콘솔의 **설정** 화면과 메타데이터 데이터베이스에 있습니다.

변수	기본값	필수	설명
`INVENQOR_LISTEN_ADDRESS`	`127.0.0.1:7070`	컨테이너에서 필수	수신 주소. 컨테이너에서는 `0.0.0.0:7070`으로 지정해야 외부에서 접속됩니다.
`INVENQOR_BASE_URL`	없음	아니오	외부에서 보이는 기본 URL. `http` / `https`만 허용하며 Keycloak Redirect URI 생성 등에 사용합니다.
`INVENQOR_STATE_DIR`	`/var/lib/invenqor-server`	아니오	상태 디렉터리. 절대 경로여야 합니다. 아래 세 경로의 기본값이 여기서 파생됩니다.
`INVENQOR_SQLITE_PATH`	`<state>/invenqor.db`	아니오	SQLite 파일 경로. PostgreSQL DSN을 지정하면 사용하지 않습니다.
`INVENQOR_UPDATE_DIR`	`<state>/updates`	아니오	게시된 Agent 업데이트 artifact 보관 경로. 멀티 Pod에서는 RWX 볼륨입니다.
`INVENQOR_EVENT_SPOOL_DIR`	`<state>/spool`	아니오	DB 장애 시 Agent 이벤트를 받아두는 spool 경로. 멀티 Pod에서는 RWX 볼륨입니다.
`INVENQOR_POSTGRES_DSN`	없음	운영 배포에서 필수	운영 PostgreSQL DSN. 지정하면 연결·migration 실패 시 SQLite로 내려가지 않고 기동과 readiness가 실패합니다. 별칭 `POSTGRES_DSN`, `postgres_dsn`.
`INVENQOR_POSTGRES_SCHEMA`	`public`	아니오	사용할 PostgreSQL 스키마 이름.
`INVENQOR_DATABASE_TIMEOUT`	`5s`	아니오	데이터베이스 작업 timeout. Go duration 표기(`750ms`, `10s`).
`INVENQOR_SHUTDOWN_TIMEOUT`	`15s`	아니오	종료 시 진행 중 요청을 기다리는 시간.
`INVENQOR_MASTER_KEY_FILE`	없음(상태 디렉터리에서 생성)	멀티 Pod에서 필수	비밀 설정을 봉인하는 32-byte Master Key 파일의 절대 경로. 모든 Pod가 같은 값을 사용해야 합니다.

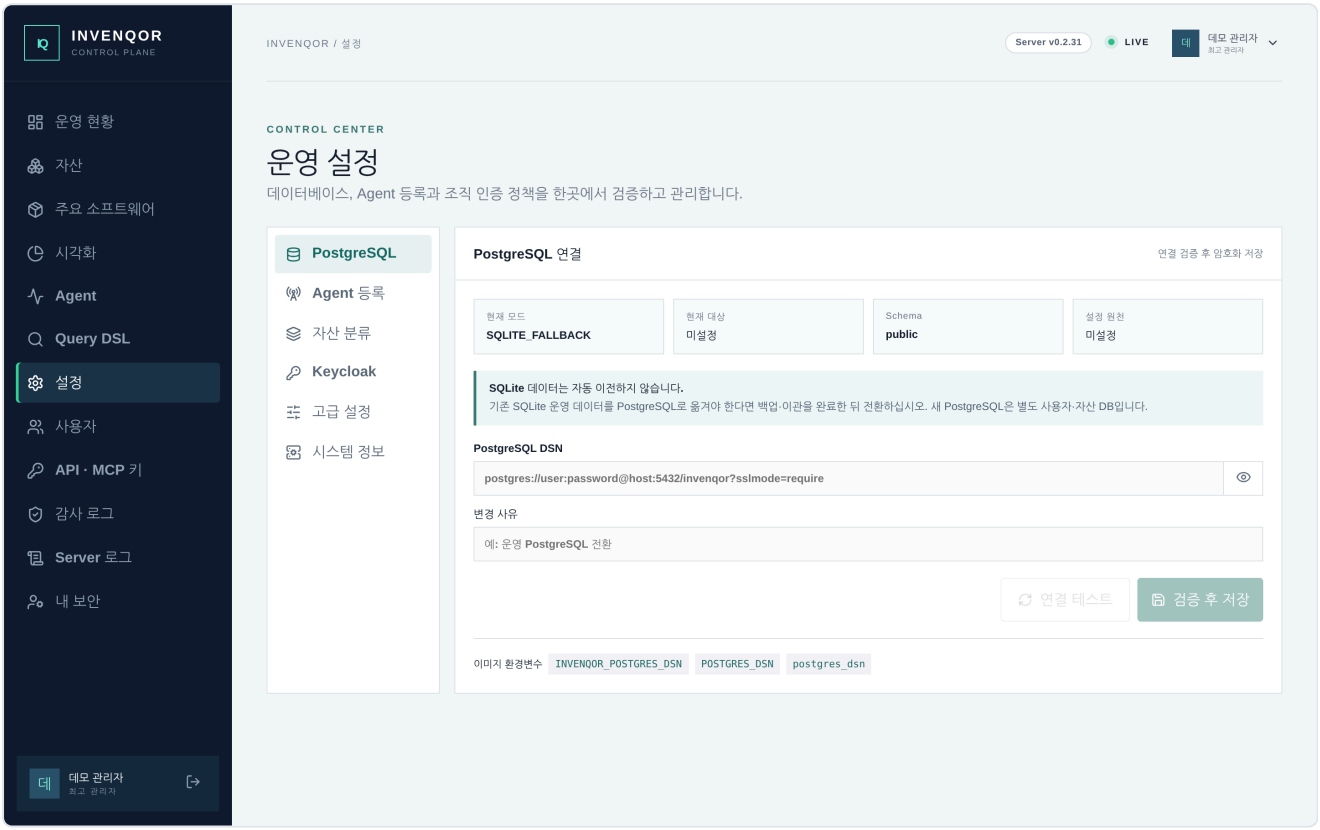
변수	기본값	필수	설명
`INVENQOR_BOOTSTRAP_ADMIN`	없음	아니오	최초 관리자 ID. 비밀번호와 반드시 함께 지정합니다. 별칭 `BOOTSTRAP_ADMIN`, `bootstrap_admin`.
`INVENQOR_BOOTSTRAP_ADMIN_PASSWORD`	없음	아니오	최초 관리자 비밀번호. 예: `ChangeMe-With-A-Strong-Password-42!`. 별칭 `BOOTSTRAP_ADMIN_PASSWORD`, `bootstrap_admin_password`.
`INVENQOR_BOOTSTRAP_ADMIN_PASSWORD_FILE`	없음	아니오	위 비밀번호를 담은 파일의 절대 경로. 값과 파일을 동시에 지정하면 기동이 거부됩니다.
`INVENQOR_AGENT_AUTO_ENROLLMENT`	`true`	아니오	최초 기동 시 자동 등록 기본 정책. 이후에는 **설정 → Agent 등록**의 DB 값이 우선합니다. 별칭 `AGENT_AUTO_ENROLLMENT`, `agent_auto_enrollment`.
`INVENQOR_AGENT_ENROLLMENT_TOKEN`	없음	아니오	등록 Token 초기값. 32자 이상이어야 하며 짧으면 기동이 실패합니다. 예: `ivq_ec_` + 임의 64자. 별칭 `AGENT_ENROLLMENT_TOKEN`, `agent_enrollment_token`.
`INVENQOR_AGENT_ENROLLMENT_TOKEN_FILE`	없음	아니오	위 Token을 담은 파일의 절대 경로.
`INVENQOR_UPDATE_PUBLIC_KEY`	없음	권장	Agent 업데이트 서명 검증용 Ed25519 공개키(base64). 지정하지 않으면 기동 시 경고를 남기고 게시 시점 검증을 건너뜁니다. 별칭 `UPDATE_PUBLIC_KEY`, `update_public_key`.
`INVENQOR_UPDATE_PUBLIC_KEY_FILE`	없음	아니오	위 공개키를 담은 파일의 절대 경로.

비밀값은 값 대신 `*_FILE` 변수와 Secret volume을 사용하십시오. 표의 예시 값은 모두 가짜이며 그대로 사용해서는 안 됩니다. 설치 절차 전체와 Kubernetes·Helm 배포는 Server 설치 및 운영 가이드를 따릅니다.

19. 운영 설정 화면

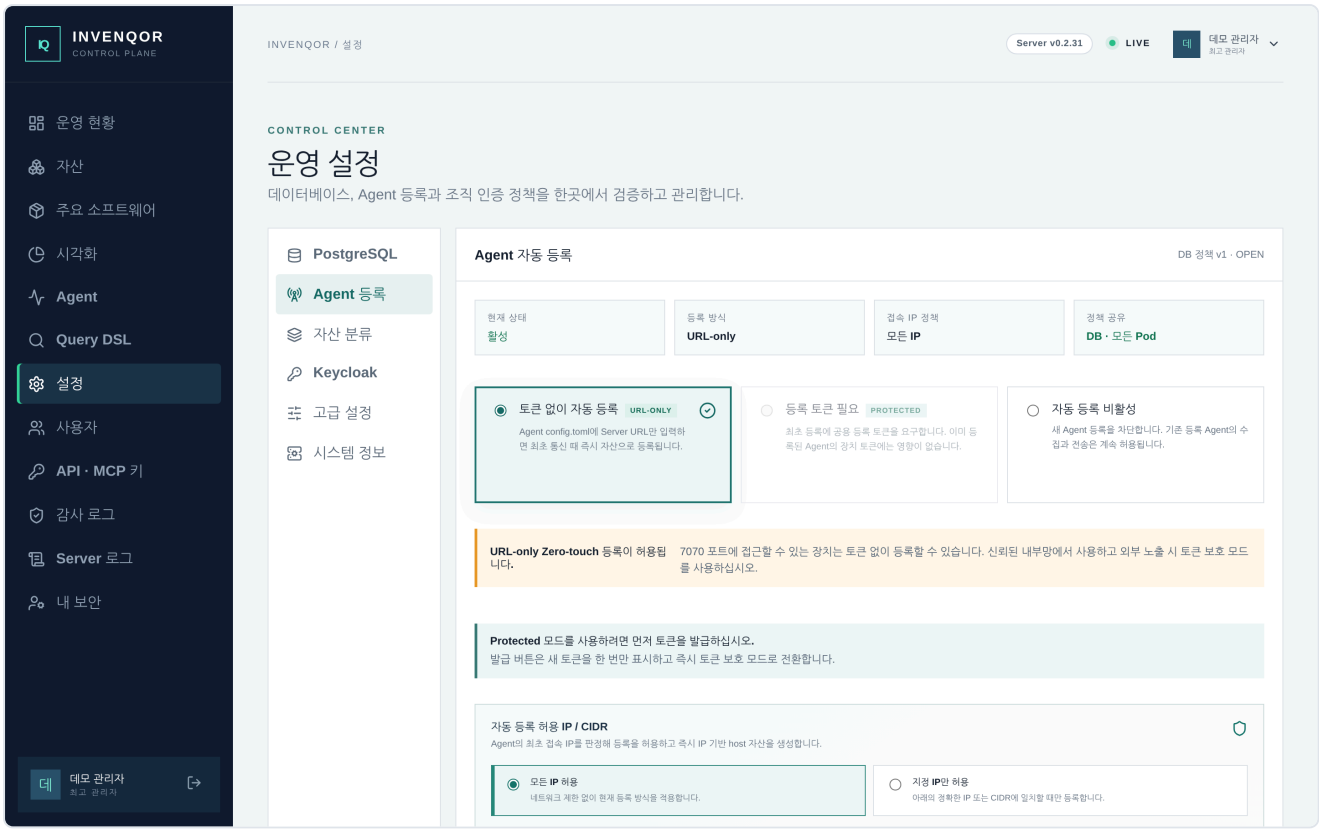
설정은 PostgreSQL, Agent 등록, 자산 분류, Keycloak, 고급 설정, 방문 추적, 시스템 정보 일곱 개 하위 화면으로 나뉩니다. 선택한 하위 화면은 `#/settings/agents` 같은 URL과 사용자별 브라우저 상태에 함께 저장되어 새로고침 후에도 유지됩니다.

PostgreSQL은 DSN을 저장하기 전에 실제 연결을 시험합니다. 현재 실행 중인 데이터베이스 모드도 같은 화면에 표시되므로, SQLite fallback으로 뜬 Pod를 여기서 가장 빨리 알아차릴 수 있습니다.



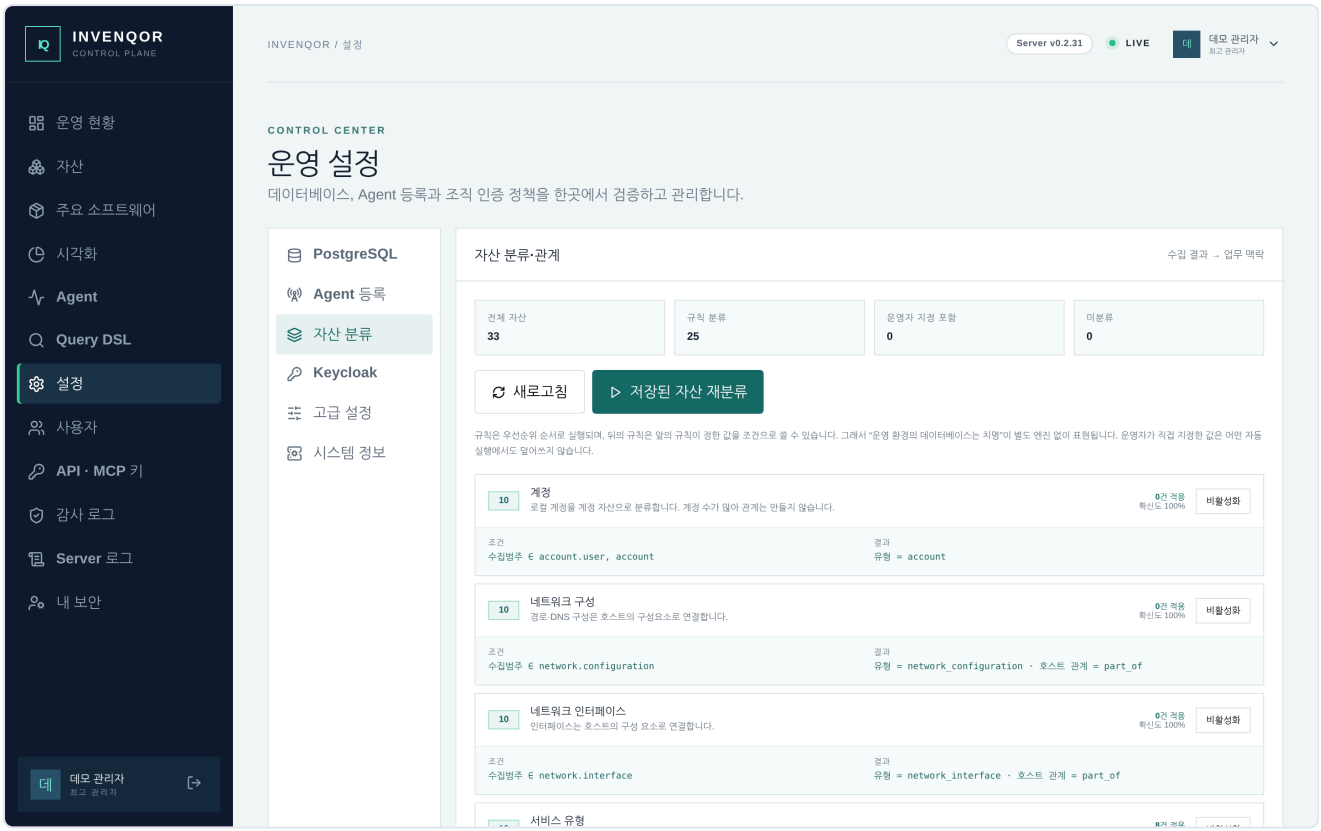
설정 → PostgreSQL — 연결을 시험한 뒤 저장하고 현재 데이터베이스 모드를 확인한다

Agent 등록은 재기동 없이 URL-only/Token 보호/차단을 전환하고 등록 Token을 발급·회전·폐기하며 IP/CIDR allowlist와 신뢰 프록시를 통제합니다. 여기서 바꾼 값이 환경 변수보다 우선합니다.



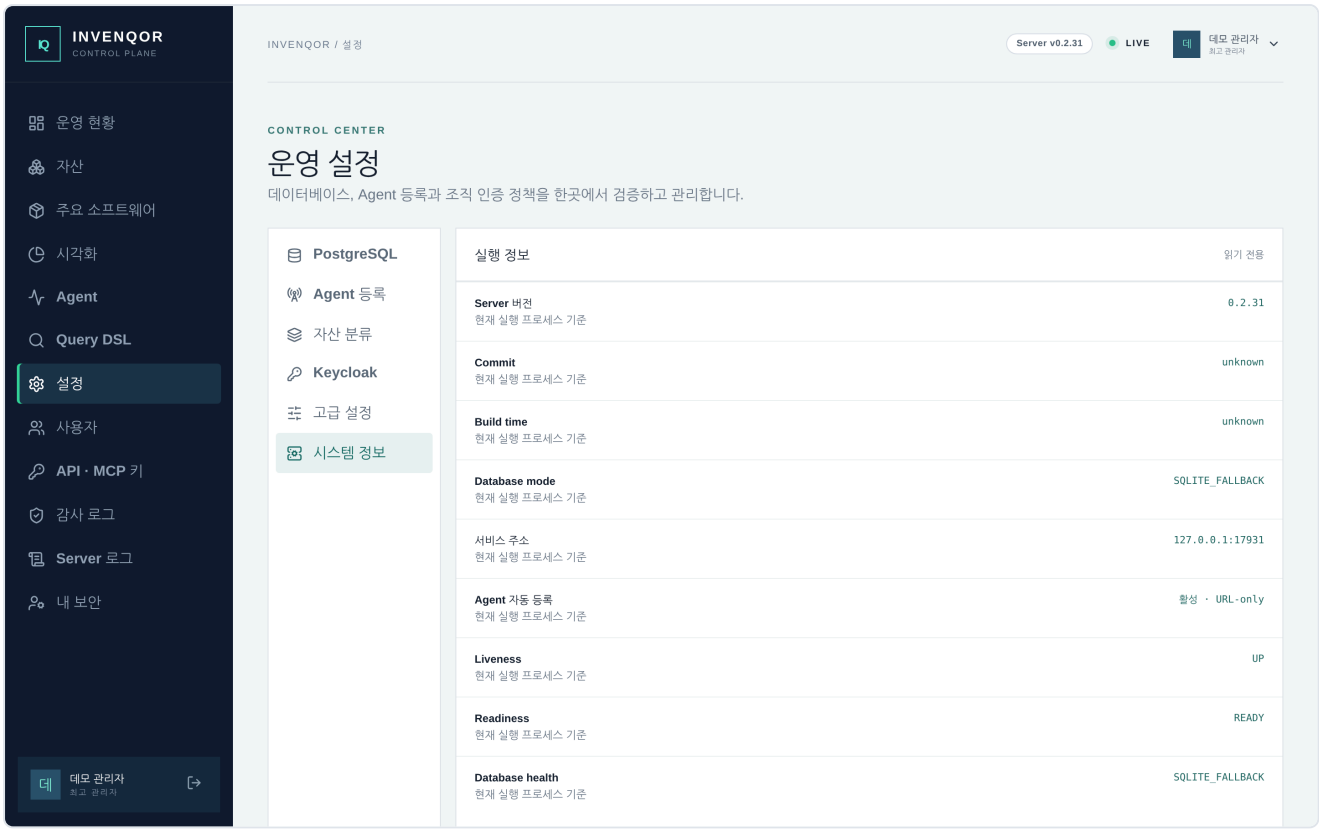
설정 → Agent 등록 — 등록 모드, 등록 Token, IP/CIDR allowlist와 신뢰 프록시를 재기동 없이 바꾼다

자산 분류는 수집 범주를 자산 유형과 호스트 관계로 옮기는 규칙을 우선순위 순서로 관리합니다. 뒤의 규칙은 앞의 규칙이 정한 값을 조건으로 쓸 수 있고, 운영자가 직접 지정한 값은 자동 실행으로 덮어쓰지 않습니다. 규칙을 바꾼 뒤에는 **저장된 자산 재분류**로 기존 자산에 다시 적용합니다.



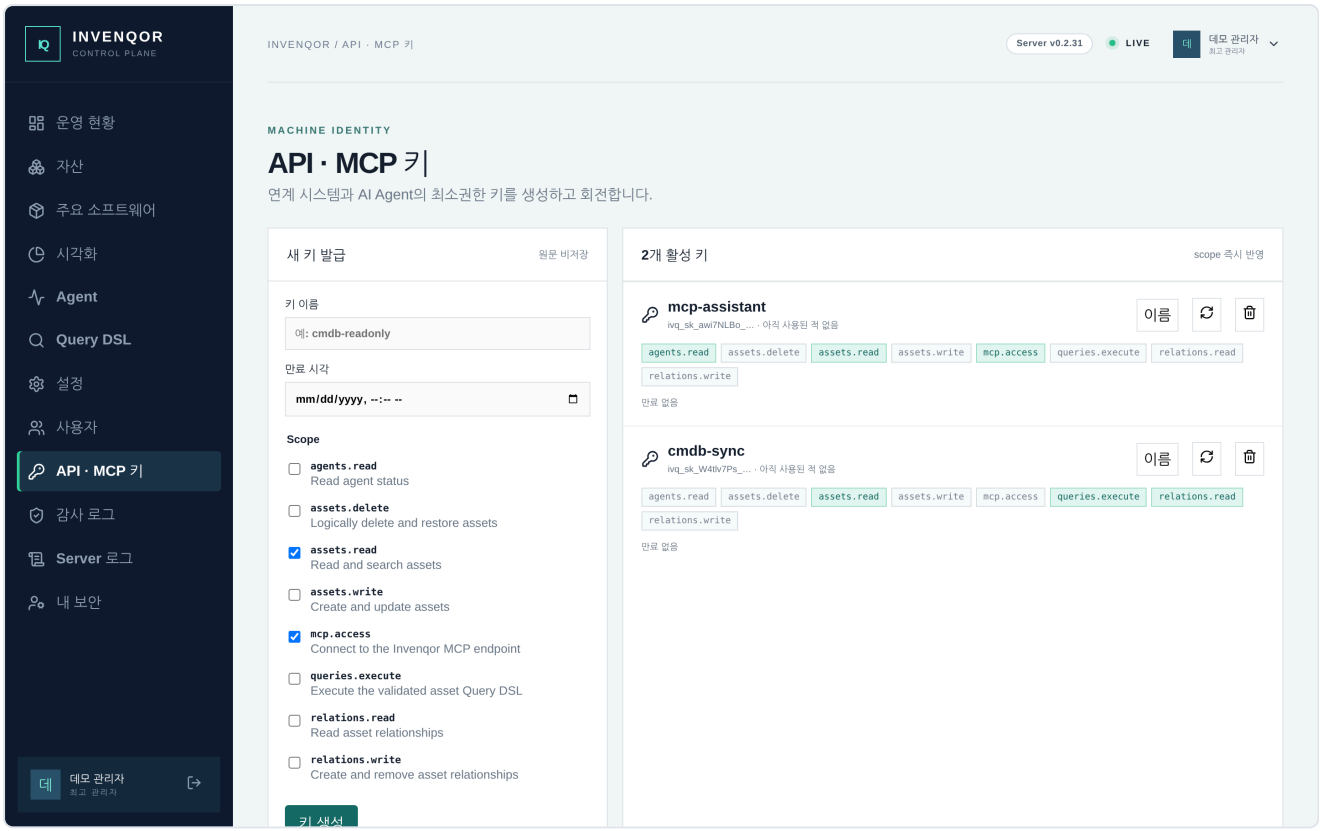
설정 → 자산 분류·관계 — 수집 범주를 자산 유형·호스트 관계로 옮기는 규칙을 우선순위 순서로 관리하고 저장된 자산에 재적용한다

시스템 정보는 읽기 전용입니다. Server 버전과 Commit·Build time, Database mode, 서비스 주소, Agent 자동 등록 상태, Liveness·Readiness·Database health를 현재 실행 프로세스 기준으로 보여 줍니다. 장애 문의 시 이 화면의 값을 그대로 전달하십시오.



설정 → 시스템 정보 — Server 버전, Database mode, Liveness·Readiness를 현재 실행 프로세스 기준으로 확인한다

API · MCP 키는 사람 Session과 분리된 기계 자격 증명을 관리합니다. scope는 필요한 것만 고르고, 원문 Secret은 발급·회전 응답에서 한 번만 표시됩니다. 수명주기 상세는 자산 API·MCP·키 관리 가이드를 따릅니다.



API · MCP 키 — 최소권한 scope로 키를 발급하고 이름 변경·회전·폐기를 수행한다

20. 방문 추적 스크립트와 CSP

설정 → 방문 추적(`#/settings/tracking`)에서 콘솔에 방문 추적 스크립트를 붙입니다. 설정은 다른 운영 정책과 같이 공용 DB(`server_metadata` 의 `tracking_policy`)에 저장되어 모든 Pod 에 즉시 적용되며, 재기동이나 재배포가 필요하지 않습니다. **기본값은 꺼짐**입니다. 새로 설치한 곳에서는 아무 스니펫도 붙지 않고 정책 헤더도 이전 버전과 바이트 단위로 같습니다.

20.1 설정 항목

항목	뜻
방문 추적 켜기	꺼짐이 기본값입니다. 켜기 전에 provider 와 그 식별자를 채워야 저장됩니다
Provider	`momento` · `ga4` · `gtm` · `matomo` · `custom` . Momento 가 첫 자리입니다
Momento 수집기 주소 · 사이트 ID · 환경	사내 Momento 수집기의 주소와 사이트 id, `data-environment` 값(기본 `prd`)
같은 오리진 프록시 사용	Momento 전용. 기본 켜짐. 아래 20.3 참조
측정 ID / 컨테이너 ID	GA4 의 `G-…` , GTM 의 `GTM-…`
Matomo 주소 · 사이트 ID	Matomo 설치 주소와 사이트 id
스니펫	`custom` 에서 받은 스니펫을 그대로 붙여 넣습니다. **8KB 를 넘으면 저장되지 않습니다**
삽입 위치	`head` 끝(기본) 또는 `body` 끝
추가 허용 출처	스니펫에서 자동으로 못 읽은 출처를 한 줄에 하나씩. `https://host` 꼴의 http(s) 출처만 받으며 `https://*.corp.example` 같은 와일드카드 라벨은 허용됩니다
변경 사유	감사 기록(`tracking.policy.update`)에 남습니다

Invenqor 콘솔은 하나의 SPA 셸(`/`)이고 화면 구분은 `#/…` 해시 경로라 Server 가 어느 화면이 열렸는지 알 수 없습니다. 그래서 다른 서비스에 있는 "관리 화면 제외" (`include_admin`) 설정은 두지 않았습니다 — 콘솔 전체가 관리 화면이므로 켜면 로그인 화면을 포함한 모든 화면에 붙습니다. 추적 도구가 URL 을 보낼 때 해시 경로가 함께 갈 수 있으니, 개인 식별 값을 보내지 않도록 도구 쪽에서 설정하십시오.

20.2 CSP 와 nonce — 스니펫이 조용히 막히지 않게 하는 방법

콘솔은 `Content-Security-Policy: default-src 'self'; …` 로 잠겨 있습니다. 이 정책 아래에서는 인라인 `<script>` 도, 외부 주소의 `<script src>` 도 브라우저가 실행하지 않으며, 그 사실은 브라우저 개발자 도구의 콘솔에만 나타납니다. 스니펫을 그냥 붙이면 화면은 멀쩡하지만 수집은 하나도 들어오지 않습니다.

추적을 켜면 Server 는 페이지 요청마다 다음을 합니다.

1. 무작위 **nonce** 를 만들어 스니펫의 모든 `<script>` 태그에 `nonce="…"` 로 붙이고, 같은 값을 `script-src 'self' 'nonce-…'` 로 정책에 넣습니다. nonce 는 요청마다 다르므로 응답을 가로채 재사용할 수 없습니다.
2. 스니펫이 부르는 **출처를 정책에 더합니다**. provider 가 정해진 것(GA4·GTM· Matomo)은 알려진 주소를, `custom` 은 붙여 넣은 문자열에서 `http(s)://…` 출처를 읽어 `script-src` · `connect-src` · `img-src` 에 넣습니다. "추가 허용 출처"의 값도 같은 세 지시어에 들어갑니다.
3. `report-uri /api/v1/tracking/csp-report` 를 정책에 넣어 브라우저가 차단한 요청을 Server 에 신고하게 합니다(20.4).

`'unsafe-inline'` 은 **어떤 경우에도 넣지 않습니다**. 한 번 넣으면 콘솔의 모든 인라인 스크립트가 함께 허용되고 추적을 끈 뒤에도 정책은 느슨한 채 남기 때문입니다. "추가 허용 출처"에 `'unsafe-inline'` 같은 키워드를 넣으면 400 으로 거절됩니다. 추적을 끄면 정책은 원래 문자열로 돌아갑니다.

`/api/*` · `/v1/*` · `/health/*` · `/mcp` · `/momento/*` 처럼 화면이 아닌 응답에는 스니펫이 붙지 않으며 정책도 `default-src 'none'; frame-ancestors 'none'` 으로 더 좁습니다.

20.3 Momento 같은 오리진 프록시

Momento 는 사내 자체 호스팅 수집기라 데이터가 밖으로 나가지 않는 유일한 선택지입니다. "같은 오리진 프록시 사용"(기본 켜짐)을 두면 스니펫은 다음과 같이 나가고,

```
<script async src="/momento/tracker.js" nonce="…"
        data-site-id="<사이트 ID>" data-environment="prd"
        data-contract-version="1" data-endpoint="/momento"></script>
```

브라우저는 Invenqor Server 의 `/momento/*` 만 부릅니다. Server 가 그 경로를 "Momento 수집기 주소"로 넘기므로 **정책에 외부 출처가 아예 등장하지 않고**, Ingress 나 사내 프록시가 브라우저에서 수집기로 가는 길을 열어 주지 않아도 됩니다. 프록시는 다음 규칙을 따릅니다.

- Momento 추적이 켜져 있을 때만 동작합니다. 그 밖에는 `/momento/*` 가 404 입니다.
- 방문자의 Invenqor 세션 쿠키와 `Authorization` 헤더는 수집기로 보내지 않고, 수집기가 돌려주는 `Set-Cookie` 는 버립니다.
- 한 번에 넘기는 본문은 256KB 까지이며, 수집기가 10초 안에 응답하지 않으면 502 로 끝냅니다. 정상 응답은 Server 로그(진단 기록)에 남기지 않고, 실패한 요청만 남습니다.
- 수집기는 `X-Forwarded-For` 로 방문자 주소를 받습니다.

프록시를 끄면 스니펫이 수집기 주소를 직접 부르고 그 출처가 세 지시어에 더해집니다. Server 에서 수집기로 가는 경로가 없고 브라우저에서 가는 경로만 있는 배치라면 이 쪽을 쓰십시오.

20.4 차단된 출처 확인과 한 번에 허용

추적이 켜져 있으면 브라우저는 정책이 막은 요청을 `POST /api/v1/tracking/csp-report` 로 신고합니다. Server 는 인증 없이 그 신고를 받아 **출처와 지시어**를 메모리에 기억합니다 — 같은 차단이 페이지마다 반복되므로 횟수가 아니라 서로 다른 출처가 중요하고, Pod 당 100개까지만 담으며 재기동하면 사라집니다. 감사 기록이 아니라 스니펫을 고치는 사람을 위한 작업 보조입니다. 8KB 를 넘는 본문은 잘라 읽고, 파싱이 안 되면 버리며, 응답은 항상 204 입니다.

방문 추적 화면 아래의 "정책이 차단한 출처"가 이 목록입니다. 항목의 **허용 목록에 추가**를 누르면 그 출처가 "추가 허용 출처"에 들어가고 (`tracking.allowed_host.add` 감사 기록), 이미 허용된 출처는 "허용됨"으로 표시됩니다. **기록 비우기**로 목록을 지운 뒤 화면을 새로 열면 여전히 막히는 것이 있는지 알 수 있습니다. 여러 Pod 를 운영하면 신고가 어느 Pod 로 갔는지에 따라 목록이 다를 수 있으니, 비어 있어도 "다시 읽기"를 몇 번 눌러 보십시오.

20.5 API

메서드·경로	권한	뜻
`GET /api/v1/admin/settings/tracking`	settings.read	현재 설정과 정책에 더해진 출처 목록
`PATCH /api/v1/admin/settings/tracking`	settings.write + CSRF	있는 필드만 바꿉니다. 잘못된 값은 400, 동시 변경은 409
`GET /api/v1/admin/settings/tracking/violations`	settings.read	이 Pod 가 받은 차단 신고
`DELETE /api/v1/admin/settings/tracking/violations`	settings.write + CSRF	차단 기록 비우기
`POST /api/v1/admin/settings/tracking/allowed-hosts`	settings.write + CSRF	`{"origin":"https://…"}` 를 허용 목록에 추가
`POST /api/v1/tracking/csp-report`	없음	브라우저의 정책 위반 신고. 항상 204
`GET · POST /momento/*`	없음	Momento 같은 오리진 프록시. 꺼져 있으면 404

```
curl -sS -b cookie.txt -H "X-CSRF-Token: $CSRF" -X PATCH \
  -H 'Content-Type: application/json' \
  https://invenqor.corp.example:7070/api/v1/admin/settings/tracking \
  -d '{"enabled":true,"provider":"momento","momento_url":"https://momento.corp.example","momento_site_id":"
```

켜고 나서 확인할 것: 콘솔을 새로 열어 응답 헤더 `Content-Security-Policy` 에 `'nonce-…'` 와 `report-uri` 가 있고, 페이지 소스의 스니펫 `<script>` 에 같은 nonce 가 붙어 있으며, Momento 관리 화면에 페이지뷰가 들어오는지 봅니다. 들어오지 않으면 20.4 의 목록을 먼저 보십시오.

21. 가이드 화면 캡처 다시 만들기

이 문서와 사용자 가이드의 화면 캡처는 저장소의 스크립트가 생성합니다.

```
node scripts/capture-guide-screenshots.mjs
```

스크립트는 임시 디렉터리에 `invenqor-server` 를 빌드해 loopback 전용 포트로 띄우고, 가공한 자산·계정·Agent를 채운 뒤 headless Chrome 으로 1440x900 에서 촬영하고, 끝나면 그 Server 와 상태 디렉터리를 삭제합니다. 대상 주소를 환경 변수로 받지 않으므로 실수로 운영 배포를 촬영하거나 그 설정을 바꿀 수 없습니다. `GUIDE_CAPTURE_PORT` 로 사용할 포트만 바꿀 수 있습니다. 촬영에 쓰는 관리자·사용자 계정의 비밀번호는 실행할 때마다 무작위로 만들어 어디에도 기록하지 않으며, 로그인 화면은 비밀번호를 입력하기 전에 촬영합니다.

PDF 는 저장소 루트에서 `./scripts/build-guide-pdfs.sh` 로 만듭니다. 공용 가이드 도구(`GUIDE_TOOL`)가 필요하며, 그림이 하나라도 없으면 실패합니다.

스크립트가 촬영하는 화면 목록, `docs/assets/guide/` 의 파일, 두 가이드가 싣는 그림은 `go test ./internal/webui/` 의 `TestGuideScreenshotsMatchCaptureScript` 가 대조하므로, 화면을 추가하거나 이름을 바꾸면 세 곳을 함께 고쳐야 합니다.

문서 오류 및 제품 문의: GitHub 저장소 · 일반 사용 절차: 사용자 가이드 · 보안 취약점 보고 절차: SECURITY.md