출입 통제 시스템: API 연결 정의서 (상세)

본 문서는 `sdac-final-architecture.md` 아키텍처를 기반으로, 관리자 UI (Frontend)와 API Gateway를 포함한 애플리케이션 계층 (Backend) 간의 통신 계약(Contract)을 상세하게 정의합니다.

1. 일반 원칙

항목	상세 내용
Base URL	`https://api.sdac.local` (폐쇄망 내부 DNS에 등록된 가상 도메인)
인증 방식 (API)	**JWT (Access Token)** • `/api/auth/login`, `/api/auth/refresh`를 **제외한** 모든 API 요청은 HTTP 헤더에 `Authorization: Bearer <token>`을 포함해야 합니다. • `API Gateway`는 이 토큰을 검증하고, 토큰 내의 권한(Role/Permission)을 확인하여 API 접근을 인가합니다.
인증 방식 (단말기)	**Device Token (Pre-Shared Key)** • `SDAC 버전(옵션 1)`의 `IP 리더기`는 `API Gateway`로 요청 시, HTTP 헤더에 `X-Device-Token: <secret-key>`를 포함하여 자신을 인증합니다.
오류 응답 (JSON)	`{"timestamp": "...", "status": 401, "error": "Unauthorized", "message": "...", "path": "..."}` 형식의 표준 오류 응답을 반환합니다.
페이지네이션	목록(List)을 반환하는 모든 `GET` API는 `page` (0-based)와 `size` (default: 20) 쿼리 파라미터를 사용하며, `Page` 객체로 응답을 래핑합니다.

2. 관리자 인증 (Auth Service 연동)

- **경로:** `/api/auth`
- **설명:** 관리자 `Admin UI`의 로그인, 로그아웃, 토큰 갱신을 처리합니다.

`POST /api/auth/login`

- **설명:** 관리자 ID/PW를 사용하여 로그인하고 토큰을 발급받습니다.
- **권한:** (없음)

Request Body: `application/json`

필드명	자료형	필수	설명
`username`	String	**Y**	관리자 로그인 ID
`password`	String	**Y**	관리자 비밀번호 (평문)

Response (200 OK): `application/json`

필드명	자료형	설명
`accessToken`	String	JWT Access Token (API 호출 시 사용)
`expiresIn`	Integer	Access Token 만료 시간(초) (예: 3600)

Response Header:

- `Set-Cookie: refreshToken=...; HttpOnly; Secure; Path=/api/auth; Max-Age=...`
- (HttpOnly 쿠키로 `Refresh Token`이 브라우저에 자동 저장됩니다.)

Response (401 Unauthorized):

- ID 또는 PW가 일치하지 않을 때 반환됩니다.

POST /api/auth/refresh

- **설명:** `Access Token`이 만료되었을 때, `Refresh Token`을 사용하여 새 `Access Token`을 발급받습니다.
- **권한:** (없음)
- **Request Header:**
  - `Cookie: refreshToken=...` (브라우저가 자동으로 전송)

Response (200 OK): `application/json`

필드명	자료형	설명
`accessToken`	String	새로 발급된 JWT Access Token
`expiresIn`	Integer	토큰 만료 시간(초)

Response (401 Unauthorized):

- `Refresh Token`이 없거나 만료(무효화)되었을 때 반환되며, 이 경우 사용자는 다시 로그인해야 합니다.

POST /api/auth/logout

- **설명:** 현재 세션을 로그아웃합니다.

- **권한:** (Access Token 필요)
- **Request Header:**
  - `Authorization: Bearer <accessToken>`
  - `Cookie: refreshToken=...`

Response (204 No Content):

- 성공적으로 로그아웃되면 빈 Body를 반환합니다.

Response Header:

- `Set-Cookie: refreshToken=; Max-Age=0` (Refresh Token 쿠키 즉시 삭제)
- **서버 동작:** `Auth Service`는 `Access Token`을 `Redis` 블랙리스트에 등록하고 `Refresh Token`을 DB(또는 Redis)에서 무효화합니다.

3. 관리자 계정 및 권한(IAM) 관리 (User Service 연동)

- **경로:** `/api/admin/iam`
- **설명:** 시스템 관리자 계정, 역할(Role), 권한(Permission)을 관리합니다. (RBAC)
- **권한:** 이 섹션의 모든 API는 `Access Token`이 필요하며, 각 엔드포인트에 명시된 특정 권한(Permission)이 필요합니다.

GET /api/admin/iam/admins

- **설명:** 관리자 계정 목록을 검색 조건과 함께 조회합니다.
- **권한:** `ADMIN_READ`

Query Parameters:

필드명	자료형	필수	설명
`page`	Integer	N	페이지 번호 (0-based)
`size`	Integer	N	페이지당 개수 (기본 20)
`username`	String	N	로그인 ID (부분 일치 검색)
`name`	String	N	관리자 이름 (부분 일치 검색)
`status`	String	N	계정 상태 (`active`, `locked`)
`department_id`	UUID	N	소속 부서 ID (정확히 일치)
`phone_number`	String	N	연락처 (부분 일치 검색)

Response (200 OK): `Page<AdminUserResponse>`

- 페이지네이션 객체(`content`, `totalPages` 등)로 래핑되어 반환됩니다.
- `AdminUserResponse` 객체는 `password_hash`가 제외된 관리자 정보입니다.

POST /api/admin/iam/admins

- **설명:** 새 시스템 관리자 계정을 생성합니다.
- **권한:** `ADMIN_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`username`	String	**Y**	로그인 ID (Unique)
`password`	String	**Y**	초기 비밀번호
`name`	String	**Y**	관리자 이름
`department_id`	UUID	**Y**	소속 부서 ID (FK)
`phone_number`	String	N	연락처

Response (201 Created): `AdminUserResponse`

- 생성된 관리자 객체 1건 (PW 제외)

Response (400 Bad Request):

- `username`이 중복될 경우 반환됩니다.

GET /api/admin/iam/admins/{id}

- **설명:** 특정 관리자의 상세 정보를 조회합니다.
- **권한:** `ADMIN_READ`

Path Variable: `id` (UUID) - 조회할 관리자의 `admin_user_id`

Response (200 OK): `AdminUserResponse`

- 관리자 객체 1건 (할당된 `roles`, `department` 객체 포함)

Response (404 Not Found):

- ID에 해당하는 관리자가 없을 경우.

GET /api/admin/iam/admins/check-username

- **설명:** 관리자 계정 ID(username)의 사용 가능 여부를 확인합니다.
- **권한:** `ADMIN_READ`

Query Parameters:

필드명	자료형	필수	설명
`username`	String	**Y**	확인할 로그인 ID

Response (200 OK): `application/json`

- `{"available": true/false}` 형태로 반환됩니다.

PUT /api/admin/iam/admins/{id}

- **설명:** 특정 관리자의 기본 정보(이름, 부서, 연락처)를 수정합니다.
- **권한:** `ADMIN_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	관리자 이름
`department_id`	UUID	**Y**	소속 부서 ID (FK)
`phone_number`	String	N	연락처

Response (200 OK): `AdminUserResponse`

- 수정된 관리자 객체 1건

PUT /api/admin/iam/admins/{id}/status

- **설명:** 관리자 계정 상태를 변경합니다 (Soft Delete).
- **권한:** `ADMIN_STATUS_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	변경할 상태 (`active`, `locked`)

Response (200 OK): `AdminUserResponse`

- 상태가 변경된 관리자 객체

POST /api/admin/iam/admins/{id}/reset-password

- **설명:** 최고 관리자가 다른 관리자의 비밀번호를 강제로 재설정합니다 (폐쇄망 방식).

- **권한:** `ADMIN_PASSWORD_RESET`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`newPassword`	String	**Y**	새로운 임시 비밀번호

Response (204 No Content):

- 성공적으로 변경됨.

PUT /api/admin/iam/me/change-password

- **설명:** 로그인한 본인이 자신의 비밀번호를 변경합니다.
- **권한:** (본인 인증, Access Token 필요)

Request Body: `application/json`

필드명	자료형	필수	설명
`currentPassword`	String	**Y**	현재 비밀번호
`newPassword`	String	**Y**	새 비밀번호

Response (204 No Content):

- 성공적으로 변경됨.

GET /api/admin/iam/roles

- **설명:** 역할(Role) 목록을 조회합니다.
- **권한:** `ROLE_READ`

Query Parameters: `page`, `size`, `name`, `status`

Response (200 OK): `Page<RoleResponse>`

- 역할 객체 목록 (페이지네이션)

POST /api/admin/iam/roles

- **설명:** 새 역할(Role)을 생성합니다. (역할의 이름, 설명 등 기본 속성은 수정할 수 없음)
- **권한:** `ROLE_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	역할 이름 (Unique, 예: `ROLE_ADMIN`)
`description`	String	N	역할 설명

Response (201 Created): `RoleResponse`

- 생성된 역할 객체

GET /api/admin/iam/roles/{id}

- **설명:** 특정 역할의 상세 정보를 조회합니다 (할당된 권한 포함).
- **권한:** `ROLE_READ`

Path Variable: `id` (UUID)

Response (200 OK): `RoleResponse`

- 역할 객체 1건 (할당된 `permissions` 목록 포함)

PUT /api/admin/iam/roles/{id}/status

- **설명:** 역할(Role)을 비활성화(Soft Delete)합니다.
- **권한:** `ROLE_DELETE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`active`, `decommissioned`

Response (200 OK): `RoleResponse`

- 상태가 변경된 역할 객체

Response (409 Conflict):

- 해당 역할이 관리자에게 할당되어 있는(unused 아님) 경우.

GET /api/admin/iam/permissions

- **설명:** 권한(Permission) 목록을 조회합니다.
- **권한:** `PERMISSION_READ`

Query Parameters: `page`, `size`, `name`, `status`

Response (200 OK): `Page<PermissionResponse>`

- 권한 객체 목록 (페이지네이션)

POST /api/admin/iam/permissions

- **설명:** 새 권한(Permission)을 생성합니다. (수정(PUT)은 불변성 원칙에 따라 제공하지 않음)
- **권한:** `PERMISSION_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	권한 이름 (Unique, 예: `USER_READ`)
`description`	String	N	권한 설명

Response (201 Created): `PermissionResponse`

- 생성된 권한 객체

GET /api/admin/iam/permissions/{id}

- **설명:** 특정 권한의 상세 정보를 조회합니다.
- **권한:** `PERMISSION_READ`

Path Variable: `id` (UUID)

Response (200 OK): `PermissionResponse`

- 권한 객체 1건

PUT /api/admin/iam/permissions/{id}/status

- **설명:** 권한(Permission)을 비활성화(Soft Delete)합니다.
- **권한:** `PERMISSION_DELETE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`active`, `decommissioned`

Response (200 OK): `PermissionResponse`

- 상태가 변경된 권한 객체

Response (409 Conflict):

- 해당 권한이 역할에 할당되어 있는(unused 아님) 경우.

GET /api/admin/iam/roles/{id}/permissions

- **설명:** 특정 역할(Role)에 할당된 권한(Permission) 목록을 조회합니다.
- **권한:** `ROLE_READ`

Path Variable: `id` (UUID)
Query Parameters: `page`, `size`

Response (200 OK): `Page<PermissionResponse>`

- 해당 역할에 할당된 권한 목록

PUT /api/admin/iam/roles/{id}/permissions

- **설명:** 특정 역할(Role)에 권한(Permission)들을 매핑(할당)합니다. (M:N 관계 갱신)
- **권한:** `ROLE_PERMISSION_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`permissionIds`	Array[UUID]	**Y**	이 역할에 할당할 권한 ID 목록

Response (200 OK): `RoleResponse`

- 권한이 갱신된 역할 객체

GET /api/admin/iam/admins/{id}/roles

- **설명:** 특정 관리자에게 할당된 역할(Role) 목록을 조회합니다.
- **권한:** `ADMIN_READ`

Path Variable: `id` (UUID)
Query Parameters: `page`, `size`

Response (200 OK): `Page<RoleResponse>`

- 해당 관리자에게 할당된 역할 목록

PUT /api/admin/iam/admins/{id}/roles

- **설명:** 특정 관리자에게 역할(Role)들을 매핑(할당)합니다. (M:N 관계 갱신)(역할 전체 교체)

- **권한:** `ADMIN_ROLE_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`roleIds`	Array[UUID]	**Y**	이 관리자에게 할당할 역할 ID 목록

Response (200 OK): `AdminUserResponse`

- 역할이 갱신된 관리자 객체

4. 출입자 및 그룹 관리 (User Service 연동)

- **경로:** `/api/admin/users`
- **설명:** 출입문(Door)을 통과하는 대상인 출입자(직원, 방문객)를 관리합니다.
- **권한:** 이 섹션의 모든 API는 `Access Token` 및 `USER_` 관련 권한이 필요합니다.

GET /api/admin/users

- **설명:** 출입자 목록을 검색 조건과 함께 조회합니다.
- **권한:** `USER_READ`

Query Parameters:

필드명	자료형	필수	설명
`page`	Integer	N	페이지 번호 (0-based)
`size`	Integer	N	페이지당 개수 (기본 20)
`name`	String	N	출입자 이름 (부분 일치 검색)
`status`	String	N	계정 상태 (`active` , `suspended` , `visitor`)
`employee_id`	String	N	사번 (정확히 일치)
`email`	String	N	이메일 (부분 일치 검색)
`phone_number`	String	N	연락처 (부분 일치 검색)

Response (200 OK): `Page<UserResponse>`

- 페이지네이션 객체(`content` , `totalPages` 등)로 래핑되어 반환됩니다.
- `UserResponse`에는 `photo_url`이 포함됩니다.

POST /api/admin/users

- **설명:** 새 출입자(직원/방문객)를 등록합니다.
- **권한:** `USER_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	출입자 이름
`department_id`	UUID	**Y**	소속 부서 ID (FK)
`status`	String	**Y**	`active`, `visitor` 등
`employee_id`	String	N	사번 (Unique)
`title`	String	N	직급
`email`	String	N	이메일 (Unique)
`phone_number`	String	N	연락처

Response (201 Created): `UserResponse`

- 생성된 출입자 객체 1건. (사진, 인증 매체는 별도 API로 등록)

GET /api/admin/users/{id}

- **설명:** 특정 출입자의 상세 정보를 조회합니다.
- **권한:** `USER_READ`

Path Variable: `id` (UUID) - `user_id`

Response (200 OK): `UserResponse`

- 출입자 객체 1건 (할당된 `credentials`, `groups`, `department` 객체 포함)

PUT /api/admin/users/{id}

- **설명:** 특정 출입자의 기본 정보를 수정합니다.
- **권한:** `USER_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	출입자 이름
`department_id`	UUID	**Y**	소속 부서 ID (FK)
`employee_id`	String	N	사번 (Unique)

필드명	자료형	필수	설명
`title`	String	N	직급
`email`	String	N	이메일 (Unique)
`phone_number`	String	N	연락처

Response (200 OK): `UserResponse`

- 수정된 출입자 객체 1건

`PUT /api/admin/users/{id}/status`

- **설명:** 출입자 계정 상태를 변경합니다 (예: 퇴사자 처리).
- **권한:** `USER_STATUS_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`active`, `suspended` (정지), `visitor`

Response (200 OK): `UserResponse`

- 상태가 변경된 출입자 객체.
- **서버 동작:** `User Service`가 `Redis` 캐시를 즉시 갱신(또는 삭제)하여 실시간 판정에 반영합니다.

`GET /api/admin/users/groups`

- **설명:** 출입자 그룹 목록을 조회합니다.
- **권한:** `USER_READ`

Query Parameters: `page`, `size`, `name`, `status`

Response (200 OK): `Page<GroupResponse>`

- 그룹 객체 목록 (페이지네이션)

`POST /api/admin/users/groups`

- **설명:** 새 출입자 그룹을 생성합니다.
- **권한:** `USER_GROUP_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	그룹 이름 (Unique)
`description`	String	N	그룹 설명 (예: '서버실 접근 가능')

Response (201 Created): `GroupResponse`

- 생성된 그룹 객체

GET /api/admin/users/groups/{id}

- **설명:** 특정 출입자 그룹의 상세 정보를 조회합니다.
- **권한:** `USER_READ`

Path Variable: `id` (UUID)

Response (200 OK): `GroupResponse`

- 그룹 객체 1건

PUT /api/admin/users/groups/{id}

- **설명:** 출입자 그룹의 메타데이터(이름, 설명)를 수정합니다.
- **권한:** `USER_GROUP_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	그룹 이름 (Unique)
`description`	String	N	그룹 설명

Response (200 OK): `GroupResponse`

- 수정된 그룹 객체

PUT /api/admin/users/groups/{id}/status

- **설명:** 출입자 그룹을 비활성화(Soft Delete)합니다.
- **권한:** `USER_GROUP_DELETE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`active`, `decommissioned`

Response (200 OK): `GroupResponse`

- 상태가 변경된 그룹 객체

Response (409 Conflict):

- 해당 그룹에 할당된 출입자가 있거나, 할당된 접근 규칙(Policy)이 있는 경우.

GET /api/admin/users/groups/{id}/users

- **설명:** 특정 출입자 그룹에 속한 멤버(출입자) 목록을 조회합니다.
- **권한:** `USER_READ`

Path Variable: `id` (UUID) - `group_id` **Query Parameters:** `page`, `size`, `name`

Response (200 OK): `Page<UserResponse>`

- 해당 그룹에 속한 출입자 목록 (페이지네이션)

GET /api/admin/users/departments

- **설명:** 부서 목록을 조회합니다.
- **권한:** `ADMIN_READ`

Query Parameters: `page`, `size`, `name`, `status`

Response (200 OK): `Page<DepartmentResponse>`

- 부서 객체 목록 (페이지네이션)

POST /api/admin/users/departments

- **설명:** 새 부서를 생성합니다.
- **권한:** `DEPARTMENT_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	부서 이름 (Unique)
`description`	String	N	부서 설명 (예: 'R&D 센터')

Response (201 Created): `DepartmentResponse`

- 생성된 부서 객체

GET /api/admin/users/departments/{id}

- **설명:** 특정 부서의 상세 정보를 조회합니다.
- **권한:** ADMIN_READ

Path Variable: id (UUID)

Response (200 OK): DepartmentResponse

- 부서 객체 1건

PUT /api/admin/users/departments/{id}

- **설명:** 부서의 메타데이터(이름, 설명)를 수정합니다.
- **권한:** DEPARTMENT_UPDATE

Path Variable: id (UUID)

Request Body: application/json

필드명	자료형	필수	설명
name	String	**Y**	부서 이름 (Unique)
description	String	N	부서 설명

Response (200 OK): DepartmentResponse

- 수정된 부서 객체

PUT /api/admin/users/departments/{id}/status

- **설명:** 부서를 비활성화(Soft Delete)합니다.
- **권한:** DEPARTMENT_DELETE

Path Variable: id (UUID)

Request Body: application/json

필드명	자료형	필수	설명
status	String	**Y**	active, decommissioned

Response (200 OK): DepartmentResponse

- 상태가 변경된 부서 객체

Response (409 Conflict):

- 해당 부서에 할당된 출입자/관리자가 있는 경우.

GET /api/admin/users/departments/{id}/users

- **설명:** 특정 부서에 속한 출입자 목록을 조회합니다.
- **권한:** `USER_READ`

Path Variable: `id` (UUID) - `department_id` **Query Parameters:** `page`, `size`, `name`

Response (200 OK): `Page<UserResponse>`

- 해당 부서에 속한 출입자 목록 (페이지네이션)

GET /api/admin/users/departments/{id}/admins

- **설명:** 특정 부서에 속한 관리자 목록을 조회합니다.
- **권한:** `ADMIN_READ`

Path Variable: `id` (UUID) - `department_id` **Query Parameters:** `page`, `size`, `name`

Response (200 OK): `Page<AdminUserResponse>`

- 해당 부서에 속한 관리자 목록 (페이지네이션)

GET /api/admin/users/{id}/groups

- **설명:** 특정 출입자에게 할당된 그룹 목록을 조회합니다.
- **권한:** `USER_READ`

Path Variable: `id` (UUID) - `user_id` **Query Parameters:** `page`, `size`

Response (200 OK): `Page<GroupResponse>`

- 해당 출입자에게 할당된 그룹 목록

PUT /api/admin/users/{id}/groups

- **설명:** 특정 출입자에게 그룹(들)을 매핑(할당)합니다. (M:N 관계 갱신)(그룹 전체 교체)
- **권한:** `USER_GROUP_UPDATE`

Path Variable: `id` (UUID) - `user_id`

Request Body: `application/json`

필드명	자료형	필수	설명
`groupIds`	Array[UUID]	**Y**	이 출입자에게 할당할 그룹 ID 목록

Response (200 OK): `UserResponse`

- 그룹이 갱신된 출입자 객체

5. 인증 매체 관리 (User Service 연동)

- **경로:** `/api/admin/credentials`
- **설명:** 특정 출입자(`user_id`)에게 속한 카드, 지문, NFC, QR 코드 등을 관리합니다.
- **권한:** `CREDENTIAL_` 관련 권한 필요.

GET /api/admin/credentials

- **설명:** 인증 매체 목록을 검색 조건과 함께 조회합니다.
- **권한:** `CREDENTIAL_READ`

Query Parameters:

필드명	자료형	필수	설명
`page`	Integer	N	
`size`	Integer	N	
`user_id`	UUID	N	소유자 ID
`type`	String	N	`card`, `nfc`, `fingerprint`, `qr`
`status`	String	N	`active`, `lost`, `expired`
`value`	String	N	카드 값 (부분 일치 검색)

Response (200 OK): `Page<CredentialResponse>`

- 인증 매체 객체 목록 (페이지네이션)

POST /api/admin/credentials

- **설명:** 특정 출입자에게 새 인증 매체를 등록합니다. (예: 새 카드 발급)
- **권한:** `CREDENTIAL_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`user_id`	UUID	**Y**	소유자 ID
`type`	String	**Y**	`card`, `nfc`, `fingerprint`, `qr`
`value`	String	**Y**	카드 값, NFC ID, 지문 ID 등 (Unique)
`status`	String	**Y**	`active`
`expires_at`	DateTime	N	만료 일시 (주로 `qr` 타입 방문객용)

Response (201 Created): `CredentialResponse`

- 생성된 인증 매체 객체.
- **서버 동작:** `User Service`가 `Redis` 캐시를 즉시 갱신(동기화)합니다.

`GET /api/admin/credentials/{id}`

- **설명:** 특정 인증 매체의 상세 정보를 조회합니다.
- **권한:** `CREDENTIAL_READ`

Path Variable: `id` (UUID) - `credential_id`

Response (200 OK): `CredentialResponse`

- 인증 매체 객체 1건 (소유자 `user` 객체 포함)

`PUT /api/admin/credentials/{id}/status`

- **설명:** 인증 매체 상태를 변경합니다 (예: 카드 분실 신고, Soft Delete).
- **권한:** `CREDENTIAL_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`active`, `lost` (분실), `expired` (만료)

Response (200 OK): `CredentialResponse`

- 상태가 변경된 인증 매체 객체.
- **서버 동작:** `User Service`가 `Redis` 캐시에서 해당 `value`를 즉시 삭제(DEL)하여 판정에 반영합니다.

6. 정책/장치 관리 (Policy Service 연동)

- **경로:** `/api/admin/policies`
- **설명:** 구역(Zone), 장치(Device), 출입문(Door), 시간표(Schedule), 접근 규칙(Rule)을 관리합니다.
- **권한:** `POLICY_` 또는 `DEVICE_` 관련 권한 필요.

`GET /api/admin/policies/zones`

- **설명:** 구역(Zone) 목록을 조회합니다.
- **권한:** `POLICY_READ`

Query Parameters:

필드명	자료형	필수	설명
`page`	Integer	N	페이지 번호 (0-based)
`size`	Integer	N	페이지당 개수 (기본 20)
`name`	String	N	구역 이름 (부분 일치 검색)
`status`	String	N	`active`, `decommissioned`

Response (200 OK): `Page<ZoneResponse>`

- 구역 객체 목록 (페이지네이션)

POST /api/admin/policies/zones

- **설명:** 새 구역을 생성합니다.
- **권한:** `POLICY_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	구역 이름 (Unique)
`description`	String	N	구역 설명

Response (201 Created): `ZoneResponse`

- 생성된 구역 객체

GET /api/admin/policies/zones/{id}

- **설명:** 특정 구역의 상세 정보를 조회합니다.
- **권한:** `POLICY_READ`

Path Variable: `id` (UUID) - `zone_id`

Response (200 OK): `ZoneResponse`

- 구역 객체 1건 (할당된 `doors` 목록 포함)

GET /api/admin/policies/zones/{id}/doors

- **설명:** 특정 구역에 속한 문(Door) 목록을 조회합니다.
- **권한:** `POLICY_READ`

Path Variable: `id` (UUID) - `zone_id` **Query Parameters:** `page`, `size`, `name`

Response (200 OK): `Page<DoorResponse>`

- 해당 구역에 속한 문 목록 (페이지네이션)

PUT /api/admin/policies/zones/{id}

- **설명:** 구역의 메타데이터(이름, 설명)를 수정합니다.
- **권한:** `POLICY_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	구역 이름 (Unique)
`description`	String	N	구역 설명

Response (200 OK): `ZoneResponse`

- 수정된 구역 객체

PUT /api/admin/policies/zones/{id}/status

- **설명:** 구역을 비활성화(Soft Delete)합니다.
- **권한:** `POLICY_DELETE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`active`, `decommissioned`

Response (200 OK): `ZoneResponse`

- 상태가 변경된 구역 객체

Response (409 Conflict):

- 해당 구역에 할당된 문(Door)이 있는 경우.

GET /api/admin/policies/devices

- **설명:** 장치(Device) 목록을 검색 조건과 함께 조회합니다.
- **권한:** `DEVICE_READ`

Query Parameters:

필드명	자료형	필수	설명
page	Integer	N	
size	Integer	N	
name	String	N	장치 이름 (부분 일치)
type	String	N	sdac_reader, acu_controller, io_controller
ip_address	String	N	IP 주소
location	String	N	설치 위치 (부분 일치)
status	String	N	online, offline, locked, decommissioned
serial_number	String	N	시리얼 번호

Response (200 OK): `Page<DeviceResponse>`

- 장치 객체 목록 (페이지네이션)

POST /api/admin/policies/devices

- **설명:** 새 장치(IP 리더기, ACU 등)를 등록합니다.
- **권한:** `DEVICE_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
name	String	**Y**	장치 이름 (예: '서버실 정문 리더기')
type	String	**Y**	sdac_reader, acu_controller 등
ip_address	String	**Y**	장치 IP 주소 (INET)
device_token	String	**Y**	sdac_reader용 비밀 키 (최초 등록 시 평문)
description	String	N	상세 설명
location	String	N	물리적 설치 위치
serial_number	String	N	하드웨어 시리얼 번호 (Unique)
firmware_version	String	N	펌웨어 버전

Response (201 Created): `DeviceResponse`

- 생성된 장치 객체. (서버는 `device_token`을 해시하여 저장)

GET /api/admin/policies/devices/{id}

- **설명:** 특정 장치의 상세 정보를 조회합니다.
- **권한:** `DEVICE_READ`

Path Variable: `id` (UUID)

Response (200 OK): `DeviceResponse`

- 장치 객체 1건 (할당된 `doors` 포함)

`PUT /api/admin/policies/devices/{id}`

- **설명:** 장치의 메타데이터(이름, IP, 위치 등)를 수정합니다.
- **권한:** `DEVICE_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	장치 이름
`ip_address`	String	**Y**	장치 IP 주소 (INET)
`description`	String	N	
`location`	String	N	
`firmware_version`	String	N	

Response (200 OK): `DeviceResponse`

- 수정된 장치 객체. (보안상 Token, Serial Number는 이 API로 수정 불가)

`POST /api/admin/policies/devices/{id}/rotate-token`

- **설명:** 장치 토큰(비밀 키)이 노출되었을 때, 강제로 재발급(Rotate)합니다.
- **권한:** `DEVICE_TOKEN_RESET`

Path Variable: `id` (UUID)

Response (200 OK): `{"newDeviceToken": "..."}`

- 새 비밀 키(평문)를 **1회만** 반환합니다. 관리자는 이 키를 장치에 수동 입력해야 합니다.
- 서버는 이 새 키의 해시(Hash)값을 DB에 갱신합니다.

`PUT /api/admin/policies/devices/{id}/status`

- **설명:** 장치 상태를 변경합니다 (Soft Delete).

- **권한:** `DEVICE_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`online`, `offline`, `locked`, `decommissioned`

Response (200 OK): `DeviceResponse`

- 상태가 변경된 장치 객체

GET /api/admin/policies/doors

- **설명:** 출입문(Door) 목록을 조회합니다.
- **권한:** `POLICY_READ`

Query Parameters: `page`, `size`, `name`, `zone_id`, `status`

Response (200 OK): `Page<DoorResponse>`

- 출입문 객체 목록 (페이지네이션)

POST /api/admin/policies/doors

- **설명:** 새 출입문을 생성합니다. (장치 할당은 별도 API로 수행)
- **권한:** `POLICY_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	출입문 이름 (예: '서버실 정문')
`zone_id`	UUID	**Y**	이 문이 속한 구역 ID (FK)
`door_config`	Object	N	`{"relock_time": 5, "held_open_time": 30}` (JSONB)

Response (201 Created): `DoorResponse`

- 생성된 출입문 객체.

GET /api/admin/policies/doors/{id}

- **설명:** 특정 출입문의 상세 정보를 조회합니다.
- **권한:** `POLICY_READ`

Path Variable: `id` (UUID)

Response (200 OK): `DoorResponse`

- 출입문 객체 1건 (할당된 `devices(IN/OUT)`, `zone` 객체 포함)

PUT /api/admin/policies/doors/{id}

- **설명:** 출입문의 메타데이터(이름, 구역, 설정)를 수정합니다.
- **권한:** `POLICY_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	출입문 이름
`zone_id`	UUID	**Y**	이 문이 속한 구역 ID (FK)
`door_config`	Object	N	`{"relock_time": 5, "held_open_time": 30}`

Response (200 OK): `DoorResponse`

- 수정된 출입문 객체

PUT /api/admin/policies/doors/{id}/status

- **설명:** 출입문 상태를 변경합니다 (Soft Delete).
- **권한:** `POLICY_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`active`, `locked` (수리 중), `decommissioned`

Response (200 OK): `DoorResponse`

- 상태가 변경된 출입문 객체

GET /api/admin/policies/doors/{id}/devices

- **설명:** 특정 출입문에 할당된 장치(Device) 목록을 조회합니다 (IN/OUT).
- **권한:** `DEVICE_READ`

Path Variable: `id` (UUID) - `door_id` **Query Parameters:** `page`, `size`

Response (200 OK): `Page<DeviceMappingResponse>`

- `{"device": {...}, "direction": "IN"}` 객체 목록

PUT /api/admin/policies/doors/{id}/devices

- **설명:** 특정 출입문에 장치(들)를 매핑(할당)합니다. (M:N 관계 갱신)
- **권한:** `DEVICE_UPDATE`

Path Variable: `id` (UUID) - `door_id`

Request Body: `application/json`

필드명	자료형	필수	설명
`devices`	Array[Object]	**Y**	할당할 장치 목록
`devices[].device_id`	UUID	**Y**	장치 ID
`devices[].direction`	String	**Y**	`IN`, `OUT`, `IO` (안티-패스백용)

Response (200 OK): `DoorResponse`

- 장치 할당이 갱신된 출입문 객체

GET /api/admin/policies/schedules

- **설명:** 시간표(Schedule) 목록을 조회합니다.
- **권한:** `POLICY_READ`

Query Parameters: `page`, `size`, `name`, `status`

Response (200 OK): `Page<ScheduleResponse>`

- 시간표 객체 목록 (페이지네이션)

POST /api/admin/policies/schedules

- **설명:** 새 시간표를 생성합니다.
- **권한:** `POLICY_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	시간표 이름 (예: '주간 근무 시간')
`rules`	Object	**Y**	`{"mon": ["09:00-18:00"], "sat": ["09:00-12:00"]}` (JSONB)

Response (201 Created): `ScheduleResponse`

- 생성된 시간표 객체

GET /api/admin/policies/schedules/{id}

- **설명:** 특정 시간표의 상세 정보를 조회합니다.
- **권한:** `POLICY_READ`

Path Variable: `id` (UUID)

Response (200 OK): `ScheduleResponse`

- 시간표 객체 1건

PUT /api/admin/policies/schedules/{id}

- **설명:** 시간표의 메타데이터(이름, 규칙)를 수정합니다.
- **권한:** `POLICY_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`name`	String	**Y**	시간표 이름
`rules`	Object	**Y**	`{"mon": ["09:00-18:00"]}` (JSONB)

Response (200 OK): `ScheduleResponse`

- 수정된 시간표 객체

PUT /api/admin/policies/schedules/{id}/status

- **설명:** 시간표를 비활성화(Soft Delete)합니다.
- **권한:** `POLICY_DELETE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`active`, `decommissioned`

Response (200 OK): `ScheduleResponse`

- 상태가 변경된 시간표 객체

Response (409 Conflict):

- 해당 시간표가 접근 규칙(Policy)에 할당되어 있는 경우.

GET /api/admin/policies/rules

- **설명:** 접근 규칙(Rule) 목록을 검색 조건과 함께 조회합니다.
- **권한:** `POLICY_READ`

Query Parameters:

필드명	자료형	필수	설명
`page`	Integer	N	
`size`	Integer	N	
`group_id`	UUID	N	출입자 그룹 ID
`zone_id`	UUID	N	구역 ID
`status`	String	N	`active`, `inactive`

Response (200 OK): `Page<AccessRuleResponse>`

- 접근 규칙 객체 목록 (페이지네이션). (`group`, `zone`, `schedule` 객체 포함)

POST /api/admin/policies/rules

- **설명:** [그룹]이 [구역]을 [시간표]에 따라 접근한다는 핵심 규칙을 생성합니다.
- **권한:** `POLICY_CREATE`

Request Body: `application/json`

필드명	자료형	필수	설명
`group_id`	UUID	**Y**	[누가] 출입자 그룹 ID (FK)
`zone_id`	UUID	**Y**	[어디를] 구역 ID (FK)
`schedule_id`	UUID	**Y**	[언제] 시간표 ID (FK)

Response (201 Created): `AccessRuleResponse`

- 생성된 접근 규칙 객체.
- **서버 동작:** `Policy Service`가 `Redis` 캐시를 즉시 갱신(동기화)합니다.

GET /api/admin/policies/rules/{id}

- **설명:** 특정 접근 규칙의 상세 정보를 조회합니다.

- **권한:** `POLICY_READ`

Path Variable: `id` (UUID) - `policy_id`

Response (200 OK): `AccessRuleResponse`

- 접근 규칙 객체 1건

`PUT /api/admin/policies/rules/{id}`

- **설명:** 접근 규칙의 매핑을 수정합니다.
- **권한:** `POLICY_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`group_id`	UUID	**Y**	[누가] 출입자 그룹 ID (FK)
`zone_id`	UUID	**Y**	[어디를] 구역 ID (FK)
`schedule_id`	UUID	**Y**	[언제] 시간표 ID (FK)

Response (200 OK): `AccessRuleResponse`

- 수정된 접근 규칙 객체.
- **서버 동작:** `Policy Service`가 `Redis` 캐시를 즉시 갱신(동기화)합니다.

`PUT /api/admin/policies/rules/{id}/status`

- **설명:** 접근 규칙을 비활성화(Soft Delete)합니다.
- **권한:** `POLICY_UPDATE`

Path Variable: `id` (UUID)

Request Body: `application/json`

필드명	자료형	필수	설명
`status`	String	**Y**	`active`, `inactive`

Response (200 OK): `AccessRuleResponse`

- 상태가 변경된 접근 규칙 객체.
- **서버 동작:** `Policy Service`가 `Redis` 캐시를 즉시 갱신(동기화)합니다.

7. 로그 및 감사 조회 (Log Service 연동)

- **경로:** `/api/admin/logs`
- **설명:** Kafka를 통해 수집된 영구 로그를 조회합니다.
- **권한:** `LOG_` 관련 권한 필요.

GET /api/admin/logs/access

- **설명:** 출입 기록을 상세 조건으로 검색합니다 (Pagination 필수).
- **권한:** `LOG_READ_ACCESS`

Query Parameters:

필드명	자료형	필수	설명
`page`	Integer	N	페이지 번호 (0-based)
`size`	Integer	N	페이지당 개수 (기본 20)
`start_dt`	DateTime	N	검색 시작 일시 (ISO 8601)
`end_dt`	DateTime	N	검색 종료 일시
`user_id`	UUID	N	특정 출입자 ID
`door_id`	UUID	N	특정 문 ID
`result`	String	N	`grant`, `deny`
`raw_credential`	String	N	원본 카드 값 (부분 일치)

Response (200 OK): `Page<AccessLogResponse>`

- 출입 로그 객체 목록 (페이지네이션)

GET /api/admin/logs/access/{log_id}

- **설명:** 특정 출입 기록(access_log) 1건의 상세 정보를 조회합니다.
- **권한:** `LOG_READ_ACCESS`

Path Variable: `log_id`

Response (200 OK): `AccessLogResponse`

GET /api/admin/logs/access/{log_id}/video-url

- **설명:** 특정 출입 기록(`log_id`)과 연동된 CCTV 영상 스트리밍 URL을 조회합니다.
- **권한:** `LOG_READ_ACCESS`

Path Variable: `log_id`

Response (200 OK): `{"videoUrl": "..."}`

- `videoUrl`: (String) NVR/VMS 서버로부터 받은 실제 영상 스트리밍 URL (예: `rtsp://...` 또는 `http://.../stream.m3u8`). Admin UI의 비디오 플레이어가 이 URL을 재생합니다.

GET /api/admin/logs/audit

- **설명:** 관리자 활동 로그를 상세 조건으로 검색합니다 (Pagination 필수).
- **권한:** `LOG_READ_AUDIT`

Query Parameters:

필드명	자료형	필수	설명
`page`	Integer	N	
`size`	Integer	N	
`start_dt`	DateTime	N	검색 시작 일시
`end_dt`	DateTime	N	검색 종료 일시
`admin_user_id`	UUID	N	특정 관리자 ID
`action_type`	String	N	활동 유형 (예: `POLICY_UPDATE`)
`source_ip`	String	N	접속 IP (INET)

Response (200 OK): `Page<AdminAuditLogResponse>`

- 관리자 활동 로그 객체 목록 (페이지네이션)

GET /api/admin/logs/audit/{log_id}

- **설명:** 특정 관리자 활동 로그(audit_log) 1건의 상세 정보를 조회합니다.
- **권한:** `LOG_READ_AUDIT`

Path Variable: `log_id`

Response (200 OK): `AdminAuditLogResponse`

- 관리자 활동 로그객체 1건 (details 포함)

GET /api/admin/logs/summary/access

- **설명:** 출입 통계 데이터를 조회합니다.
- **권한:** `LOG_READ_ACCESS`

Query Parameters:

필드명	자료형	필수	설명
`date_range`	String	**Y**	`daily`, `monthly`

필드명	자료형	필수	설명
`group_by`	String	N	`zone`, `user`, `result`

Response (200 OK): `Object`

- `Log Service`가 DB에서 집계(Aggregation)한 통계 JSON 객체.

8. 실시간 통신 (WebFlux SSE 연동)

- **방식:** 서버가 WebFlux의 Server-Sent Events (SSE)를 사용하여 클라이언트로 이벤트를 Push합니다.
- **접속:** Admin UI는 `EventSource` API를 사용하여 API Gateway를 통해 SSE 엔드포인트에 연결합니다.
- **MIME Type:** `text/event-stream`

GET /api/admin/events/stream

- **설명:** 실시간 이벤트(출입, 장치 상태, 알람) 스트림을 구독합니다.
- **권한:** (Access Token 필요)
- **Response (200 OK):**
  - 클라이언트로 `text/event-stream`이 지속적으로 전송됩니다.
  - **Event:** `live_access_event`
  - **Data:** `{"user_name": "...", "door_name": "...", "timestamp": "...", "result": "GRANT"}`
  - **Event:** `device_status_event`
  - **Data:** `{"device_id": "...", "name": "...", "status": "OFFLINE"}`
  - **Event:** `alarm_event`
  - **Data:** `{"type": "DOOR_FORCED_OPEN", "door_name": "...", "timestamp": "..."}`
- **서버 동작:** `Log Service`가 Kafka의 `access_events`, `device_status_events`, `alarm_events` 토픽을 구독(Consume)하고, 수신된 메시지를 WebFlux를 통해 SSE 이벤트로 변환하여 클라이언트에 Push합니다.

9. 파일 업로드 (MinIO 연동)

- **설명:** 출입자/관리자 사진 업로드를 위한 2단계 프로세스입니다.
- **권한:** `FILE_UPLOAD` 권한 필요.

POST /api/admin/uploads/prepare

- **설명:** (1단계) 실제 파일 업로드 전에, 파일을 업로드할 임시 URL을 `MinIO`로부터 발급받습니다.
- **권한:** `FILE_UPLOAD`

Request Body: `application/json`

필드명	자료형	필수	설명
`filename`	String	**Y**	원본 파일명 (예: `profile.jpg`)
`content_type`	String	**Y**	파일 MIME 타입 (예: `image/jpeg`)

Response (200 OK): `{"upload_url": "...", "file_key": "..."}`

- `upload_url`: (String) `Admin UI`가 이 URL로 `PUT` 요청을 보내 파일을 업로드합니다.
- `file_key`: (String) 업로드 완료 보고 시 사용할 파일 식별자.

POST /api/admin/uploads/complete

- **설명:** (2단계) `MinIO`에 파일 업로드가 완료된 후, 해당 파일이 누구의 것인지 `User Service`에 보고합니다.
- **권한:** `FILE_UPLOAD`

Request Body: `application/json`

필드명	자료형	필수	설명
`target_type`	String	**Y**	`user` (출입자) 또는 `admin` (관리자)
`target_id`	UUID	**Y**	`user_id` 또는 `admin_user_id`
`file_key`	String	**Y**	`prepare` 단계에서 받은 파일 식별자

Response (200 OK): `UserResponse` 또는 `AdminUserResponse`

- `photo_url`이 갱신된 객체.

10. 비상/명령 API (Access Control Service 연동)

- **경로:** `/api/admin/commands`
- **설명:** 관리자가 `Admin UI`에서 수동으로 장치를 제어합니다. (높은 권한 필요)

POST /api/admin/commands/open-door

- **설명:** 특정 문을 원격으로 개방합니다.

- **권한:** `COMMAND_DOOR_OPEN`

Request Body: `application/json`

필드명	자료형	필수	설명
`door_id`	UUID	**Y**	개방할 출입문 ID
`reason`	String	**Y**	원격 개방 사유 (감사 로그용)

Response (202 Accepted):

- 명령이 수신되었음을 의미 (즉각적인 처리를 보장하지는 않음)
- **서버 동작:** `Access Control Service`가 `IP 리더기` / `ACU`에 개방 명령을 전송하고, 이 행위를 `Kafka` 감사 로그로 발행합니다.

POST /api/admin/commands/lockdown

- **설명:** 특정 구역(또는 전체)을 즉시 봉쇄합니다.
- **권한:** `COMMAND_LOCKDOWN`

Request Body: `application/json`

필드명	자료형	필수	설명
`zone_id`	UUID	N	특정 구역 ID (없을 경우 전체 시스템)
`status`	Boolean	**Y**	`true` (봉쇄 시작), `false` (봉쇄 해제)

Response (200 OK):

- **서버 동작:** `Access Control Service`가 `Redis`의 전역 상태 키(예: `global:lockdown:zone_A`)를 `true`로 설정하여, 모든 판정 로직이 이를 최우선으로 확인하게 합니다.

POST /api/admin/commands/all-open

- **설명:** 특정 구역(또는 전체)을 즉시 개방합니다 (화재 시와 유사).
- **권한:** `COMMAND_ALL_OPEN`

Request Body: `application/json`

필드명	자료형	필수	설명
`zone_id`	UUID	N	특정 구역 ID (없을 경우 전체 시스템)
`status`	Boolean	**Y**	`true` (개방 시작), `false` (개방 해제)

Response (200 OK):

- **서버 동작:** `Access Control Service`가 `Redis`의 전역 상태 키(예: `global:all-open:zone_A`)를 `true`로 설정합니다.

11. 단말기 API (SDAC 버전 Pull 방식)

- **경로:** `/api/access`
- **설명:** `IP 리더기(Dumb Terminal)`가 중앙 서버에 실시간 판정을 요청하는 엔드포인트입니다. `X-Device-Token` 헤더로 인증됩니다.

POST /api/access/attempt

- **설명:** 실시간 출입 판정을 요청합니다.
- **권한:** (Device Token 인증)

Request Body: `application/json`

필드명	자료형	필수	설명
`credential_type`	String	**Y**	`card`, `nfc`, `fingerprint`, `qr`
`credential_value`	String	**Y**	리더기가 읽은 ID 값

Response (200 OK): `application/json`

필드명	자료형	설명
`result`	String	`GRANT` 또는 `DENY`
`reason`	String	거부 사유 (예: `POLICY_TIME`, `INVALID_CARD`)
`user_name`	String	출입자 이름 (문 열림 시 표시용)

- **서버 동작:** `API Gateway`가 `Device Token` 검증 후 `Access Control Service`로 라우팅. 0.1초 내 `Redis` 캐시 조회 후 응답.

POST /api/access/event

- **설명:** 단말기(I/O 컨트롤러, 리더기)가 도어 센서 상태나 알람 이벤트를 비동기적으로 보고합니다.
- **권한:** (Device Token 인증)

Request Body: `application/json`

필드명	자료형	필수	설명
`event_type`	String	**Y**	`DOOR_SENSOR_OPENED`, `DOOR_SENSOR_CLOSED`, `DOOR_FORCED_OPEN`
`timestamp`	DateTime	**Y**	이벤트 발생 시간

Response (200 OK):

- 서버가 이벤트를 수신했음을 의미.
- **서버 동작:** `API Gateway`가 `Access Control Service`로 라우팅. `Access Control Service`는 이 이벤트를 받아 "문 열림 시간 초과" 타이머를 시작/중지하거나 "강제 개방" 알람을 `Kafka`로 발행합니다.

12. 비상 API (I/O 컨트롤러 연동)

- **경로:** `/api/emergency`
- **설명:** 화재 수신반(FACP) 등 외부 I/O 컨트롤러가 호출하는 최우선 비상 API입니다.
- **인증:** `X-Device-Token: <secret-key>` 헤더 필요.

POST /api/emergency/fire

- **설명:** 화재 신호를 수신하여 시스템을 즉시 '전체 개방' 모드로 전환합니다.
- **권한:** (Device Token 인증)
- **Request Body:** `application/json`

필드명	자료형	필수	설명
`signal_status`	String	**Y**	`ACTIVE` (화재 발생), `CLEAR` (화재 해제)
`source`	String	N	신호 발생 위치 (예: `FACP-A`

- **Response (200 OK):** (서버가 신호를 수신했음을 의미)
- **서버 동작:** Access Control Service가 이 신호를 받아 즉시 Redis 전역 상태를 `all-open`으로 설정합니다.