출입 통제 시스템: API 연결 정의서

본 문서는 **관리자 UI(Frontend)**와 **API Gateway(Backend)** 간의 모든 통신 계약(Contract)을 상세하게 정의합니다.

1. 일반 원칙 및 인증

- **Base URL:** 모든 API 요청은 `API Gateway`의 단일 도메인(예: `https://api.your-company.com`)을 통해 이루어집니다.
- **Data Format:** 모든 요청(Request)과 응답(Response)의 Body는 `application/json` 형식을 사용합니다.
- **인증 (Authentication):**
  - `/auth/**` 경로를 제외한 **모든 API 요청**은 `Authorization: Bearer <accessToken>` 헤더에 유효한 JWT Access Token을 포함해야 합니다. (`/api/auth/logout` 제외: 로그아웃시 Access Token 필요)
  - Access Token 만료 시(HTTP 401), 프론트엔드는 `/auth/refresh` 엔드포인트를 호출하여 새 토큰을 발급받아야 합니다.
- **인가 (Authorization):**
  - `API Gateway`는 토큰의 `roles` 및 `permissions` (예: `POLICY_UPDATE`)를 검사하여, 관리자가 해당 API를 호출할 권한이 있는지 확인합니다(RBAC).
- **오류 응답:** 표준 HTTP 상태 코드를 사용합니다.
  - `400 Bad Request`: 요청 파라미터가 잘못됨.
  - `401 Unauthorized`: Access Token이 없거나 만료됨.
  - `403 Forbidden`: Access Token은 유효하나, 해당 API를 실행할 권한(Role)이 없음.
  - `404 Not Found`: 요청한 리소스(예: user_id)가 없음.
  - `500 Internal Server Error`: 백엔드 MSA 서비스 오류.

2. 관리자 인증 (Auth Service 연동)

- **경로:** `/api/auth`
- **인증:** 이 경로의 API는 `Access Token`이 필요하지 않습니다. (`/logout` 제외)

목적	엔드포인트	메소드	요청 Body (JSON)	응답 (JSON / Cookie)
관리자 로그인	`/api/auth/login`	POST	`{"username": "admin", "password": "..."}`	`{"accessToken": "...", "expiresIn": 3600}` **Cookie:** `refreshToken=...` (HttpOnly)
토큰 갱신	`/api/auth/refresh`	POST	(없음)	`{"accessToken": "...", "expiresIn": 3600}`
로그아웃	`/api/auth/logout`	POST	(없음)	(204 No Content)

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

- **경로:** `/api/admin/iam`

목적	엔드포인트	메소드	요청 Body (JSON)	비고
관리자 목록 조회	`/api/admin/iam/admins`	GET	(Query: `page`, `size`, `username`, `name`, `status`, `phone_number`)	`admin_users` 테이블 목록 (PW 제외)
관리자 상세 조회	`/api/admin/iam/admins/{id}`	GET	(없음)	`admin_users` 상세 정보 (PW 제외)
관리자 생성	`/api/admin/iam/admins`	POST	`{"username": "...", "password": "...", "name": "...",`	생성된 관리자 정보 반환

목적	엔드포인트	메소드	요청 Body (JSON)	비고
			`"department_id": "...", "phone_number": "..."}`	
관리자 정보 수정	`/api/admin/iam/admins/{id}`	PUT	`{"name": "...", "department_id": "...", "phone_number": "..."}`	**(상태 변경과 분리됨)**
관리자 상태 변경	`/api/admin/iam/admins/{id}/status`	PUT	`{"status": "locked"}`	`locked`, `active`로 계정 비활성화/활성화.
아이디 중복 확인	`/api/admin/iam/admins/check-username`	GET	`{"username": "..."}`	관리자 계정 ID(username)의 사용 가능 여부를 확인합니다.
비밀번호 강제 재설정	`/api/admin/iam/admins/{id}/reset-password`	POST	`{"newPassword": "..."}`	**(폐쇄망 방식)** 최고 관리자가 강제 재설정
내 비밀번호 변경	`/api/admin/iam/me/change-password`	PUT	`{"currentPassword": "...", "newPassword": "..."}`	로그인한 본인이 직접 변경
역할(Role) 목록 조회	`/api/admin/iam/roles`	GET	(Query: `page`, `size`, `name`, `description`)	`roles` 테이블 목록
역할 상세 조회	`/api/admin/iam/roles/{id}`	GET	(없음)	`roles` 상세 정보 (할당된 권한 포함)
역할 생성	`/api/admin/iam/roles`	POST	`{"name": "...", "description": "..."}`	생성된 역할 정보 반환 (수정(PUT)은 불가능. 불변성 원칙)
역할 삭제	`/api/admin/iam/roles/{id}`	DELETE	(없음)	**비고:** 해당 역할이 관리자에게 할당되지 않은(unused) 경우에만 삭제 가능 (409 Conflict).
권한 (Permission) 목록	`/api/admin/iam/permissions`	GET	(Query: `page`, `size`, `name`, `description`)	`permissions` 테이블 목록
권한 상세 조회	`/api/admin/iam/permissions/{id}`	GET	(없음)	`permissions` 상세 정보
권한 생성	`/api/admin/iam/permissions`	POST/PUT	`{"name": "...", "description": "..."}`	생성된 권한 정보 반환 (수정(PUT)은 불가능. 불변성 원칙)
권한 삭제	`/api/admin/iam/permissions/{id}`	DELETE	(없음)	**비고:** 해당 권한이 역할에 할당되지 않은(unused) 경우에만 삭제 가능 (409 Conflict).
역할에 권한 할당	`/api/admin/iam/roles/{id}/permissions`	PUT	`{"permissionIds": ["...", "..."]}`	`role_permissions` 테이블 갱신
관리자에 역할 할당	`/api/admin/iam/admins/{id}/roles`	PUT	`{"roleIds": ["...", "..."]}`	`admin_user_roles` 테이블 갱신

4. 출입자 및 그룹 관리 (User Service 연동)

- **경로:** `/api/admin/users`, `/api/admin/groups`

목적	엔드포인트	메소드	요청 Body (JSON)	비고
출입자 목록 조회	`/api/admin/users`	GET	(Query: `page`, `size`, `name`, `status`, `employee_id`, `email`, `phone_number`)	`users` 테이블 목록 (photo_url 포함)

목적	엔드포인트	메소드	요청 Body (JSON)	비고
출입자 생성	`/api/admin/users`	POST	`{"name": "...", "department_id": "...", "status": "active", "employee_id": "...", "title": "...", "email": "...", "phone_number": "..."}`	생성된 `user_id` 반환. (사진은 업로드 API로 별도 추가)
출입자 상세 조회	`/api/admin/users/{id}`	GET	(없음)	`users` 객체 1건 (photo_url, credentials, groups, department 포함)
출입자 정보 수정	`/api/admin/users/{id}`	PUT	`{"name": "...", "department_id": "...", "employee_id": "...", "title": "...", "email": "...", "phone_number": "..."}`	(상태 변경과 분리됨)
출입자 상태 변경	`/api/admin/users/{id}/status`	PUT	`{"status": "active" | "suspended" | "visitor"}`	출입자 계정 상태 변경 (예: 퇴사자)
출입자 그룹 목록	`/api/admin/users/groups`	GET	(Query: `page`, `size`, `name`, `description`)	`groups` 테이블 목록
출입자 그룹 생성	`/api/admin/users/groups`	POST	`{"name": "...", "description": "..."}`	
출입자 그룹 상세	`/api/admin/users/groups/{id}`	GET	(없음)	`groups` 객체 1건
출입자 그룹 수정	`/api/admin/users/groups/{id}`	PUT	`{"name": "...", "description": "..."}`	
출입자 그룹 삭제	`/api/admin/users/groups/{id}`	DELETE	(없음)	비고: 할당된 출입자가 없을 때만 삭제 가능 (409 Conflict)
그룹 멤버 조회	`/api/admin/users/groups/{id}/users`	GET	(Query: `page`, `size`, `name`)	해당 `group_id`에 속한 `users` 테이블 목록
부서 목록	`/api/admin/users/departments`	GET	(Query: `page`, `size`, `name`, `description`)	`departments` 테이블 목록
부서 생성	`/api/admin/users/departments`	POST	`{"name": "...", "description": "..."}`	
부서 상세	`/api/admin/users/departments/{id}`	GET	(없음)	`departments` 객체 1건
부서 수정	`/api/admin/users/departments/{id}`	PUT	`{"name": "...", "description": "..."}`	
부서 삭제	`/api/admin/users/departments/{id}`	DELETE	(없음)	비고: 할당된 출입자/관리자가 없을 때만 삭제 가능 (409 Conflict)
부서 출입자 조회	`/api/admin/users/departments/{id}/users`	GET	(Query: `page`, `size`, `name`)	해당 `department_id`에 속한 `users` 테이블 목록
부서 관리자 조회	`/api/admin/users/departments/{id}/admins`	GET	(Query: `page`, `size`, `name`)	해당 `department_id`에 속한 `admin_users` 테이블 목록
출입자의 그룹 조회	`/api/admin/users/{id}/groups`	GET	(없음)	해당 출입자에 할당된 `groups` 목록
출입자에 그룹 할당	`/api/admin/users/{id}/groups`	PUT	`{"groupIds": ["...", "..."]}`	`user_groups` 테이블 갱신 (M:N)

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.

목적	엔드포인트	메소드	요청 Body (JSON)	비고
인증 매체 목록	`/api/admin/credentials`	GET	(Query: `page`, `size`, `user_id`, `type`, `status`)	`credentials` 테이블 목록
인증 매체 상세	`/api/admin/credentials/{id}`	GET	(없음)	`credentials` 객체 1건
인증 매체 등록	`/api/admin/credentials`	POST	`{"user_id": "...", "type": "card", "value": "...", "status": "active"}`	`User Service`가 `Redis`에 즉시 캐시 동기화
인증 매체 상태 변경	`/api/admin/credentials/{id}/status`	PUT	`{"status": "lost" | "expired" | "active"}`	(예: 카드 분실 신고) `Redis`에 즉시 동기화

6. 정책/장치 관리 (Policy Service 연동)

- **경로:** `/api/admin/policies`
- **설명:** 구역(Zone), 장치(Device), 시간표(Schedule), 접근 규칙(Rule)을 관리합니다.

목적	엔드포인트	메소드	요청 Body (JSON)	비고
구역(Zone) 목록	`/api/admin/policies/zones`	GET	(Query: `page`, `size`, `name`, `description`, `status`)	`zones` 테이블 목록
구역 상세	`/api/admin/policies/zones/{id}`	GET	(없음)	`zones` 객체 1건
구역 생성	`/api/admin/policies/zones`	POST	`{"name": "...", "description": "..."}`	
구역 수정	`/api/admin/policies/zones/{id}`	PUT	`{"name": "...", "description": "..."}`	
구역 상태 변경(삭제)	`/api/admin/policies/zones/{id}/status`	PUT	`{"status": "active" | "decommissioned"}`	Soft Delete. 비고: 할당된 문(Door)이 없을 때만 변경 가능 (409 Conflict)
구역에 속한 문(Door) 조회	`/api/admin/policies/zones/{id}/doors`	GET	(Query: `page`, `size`, `name`)	해당 `zone_id`에 속한 `doors` 테이블 목록
장치(Device) 목록	`/api/admin/policies/devices`	GET	(Query: `page`, `size`, `name`, `type`, `sdac_reader`, `ip_address`, `description`, `location`, `serial_number`, `firmware_version`)	`devices` 테이블 목록
장치 상세	`/api/admin/policies/devices/{id}`	GET	(없음)	`devices` 객체 1건
장치 등록	`/api/admin/policies/devices`	POST	`{"name": "...", "type": "sdac_reader", "ip_address": "...", "device_token": "...", "description": "...", "location": "...", "serial_number": "...", "firmware_version": "..."}`	`device_token`은 해시(Hash)되어 저장됨
장치 수정	`/api/admin/policies/devices/{id}`	PUT	`{"name": "...", "ip_address": "...", "description": "...", "location": "...", "firmware_version": "..."}`	(Token, Serial Number 등은 수정 불가)
장치 상태 변경(삭제)	`/api/admin/policies/devices/{id}/status`	PUT	`{"status": "online" | "offline" | "locked" | "decommissioned"}`	Soft Delete. (예: 해킹 의심 시 'locked')
장치 토큰 재발급	`/api/admin/policies/devices/{id}/rotate-token`	POST	(없음)	(보안) `device_token_hash`를 갱신하고 새 토큰 1회 반환.

목적	엔드포인트	메소드	요청 Body (JSON)	비고
출입문(Door) 목록	`/api/admin/policies/doors`	GET	(Query: `page`, `size`, `name`, `zone_id`, `status`)	`doors` 테이블 목록
출입문 상세	`/api/admin/policies/doors/{id}`	GET	(없음)	`doors` 객체 1건
출입문 생성	`/api/admin/policies/doors`	POST	`{"name": "...", "zone_id": "...", "door_config": {...}}`	`door_config`: (relock_time, held_open_time)
출입문 수정	`/api/admin/policies/doors/{id}`	PUT	`{"name": "...", "zone_id": "...", "door_config": {...}}`	
출입문 상태 변경 (삭제)	`/api/admin/policies/doors/{id}/status`	PUT	`{"status": "active" | "locked" | "decommissioned"}`	Soft Delete. (예: 수리 중 'locked')
출입문의 장치 조회	`/api/admin/policies/doors/{id}/devices`	GET	(Query: `page`, `size`)	
출입문에 장치 할당	`/api/admin/policies/doors/{id}/devices`	PUT	`{"devices": [{"device_id": "...", "direction": "IN"}, ...]}`	`door_devices` M:N 매핑 테이블 갱신
시간표 (Schedule) 목록	`/api/admin/policies/schedules`	GET	(Query: `page`, `size`, `name`, `status`)	`time_schedules` 테이블 목록
시간표 상세	`/api/admin/policies/schedules/{id}`	GET	(없음)	`time_schedules` 객체 1건
시간표 생성	`/api/admin/policies/schedules`	POST	`{"name": "...", "rules": {"mon": ["09:00-18:00"], ...}}`	
시간표 수정	`/api/admin/policies/schedules/{id}`	PUT	`{"name": "...", "rules": {"mon": ["09:00-18:00"], ...}}`	
시간표 상태 변경 (삭제)	`/api/admin/policies/schedules/{id}/status`	PUT	`{"status": "active" | "decommissioned"}`	Soft Delete. 비고: 할당된 정책(Rule)이 없을 때만 변경 가능 (409 Conflict)
접근 규칙(Rule) 목록	`/api/admin/policies/rules`	GET	(Query: `page`, `size`, `group_id`, `zone_id`, `status`)	`access_policies` 테이블 목록
접근 규칙 상세	`/api/admin/policies/rules/{id}`	GET	(없음)	`access_policies` 객체 1건
접근 규칙 생성	`/api/admin/policies/rules`	POST	`{"group_id": "...", "zone_id": "...", "schedule_id": "..."}`	핵심 정책 매핑. `Redis` 캐시 동기화.
접근 규칙 수정	`/api/admin/policies/rules/{id}`	PUT	`{"group_id": "...", "zone_id": "...", "schedule_id": "..."}`	`Redis` 캐시 동기화.
접근 규칙 상태 변경	`/api/admin/policies/rules/{id}/status`	PUT	`{"status": "active" | "inactive"}`	Soft Delete. `Redis` 캐시 동기화.

7. 로그 및 감사 조회 (Log Service 연동)

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

목적	엔드포인트	메소드	요청 Body (JSON)	비고
출입 기록 검색	`/api/admin/logs/access`	GET	(Query: `page`, `size`, `start_dt`, `end_dt`, `user_id`, `door_id`, `result`, `raw_credential`)	`access_logs` 테이블 검색 (Pagination 필수)
단일 출입 기록 조회	`/api/admin/logs/access/{log_id}`	GET	(없음)	단일 `access_log` 객체의 모든 상세 정보(결과, 사

목적	엔드포인트	메소드	요청 Body (JSON)	비고
				유, 사용자, 문) 반환.
관리자 활동 검색	`/api/admin/logs/audit`	GET	(Query: `page`, `size`, `start_dt`, `end_dt`, `admin_user_id`, `action_type`, `source_ip`)	`admin_audit_logs` 테이블 검색 (Pagination 필수)
단일 관리자 활동 조회	`/api/admin/logs/audit/{log_id}`	GET	(없음)	단일 `admin_audit_log` 객체의 모든 상세 정보(변경 내역 details 포함) 반환.
출입 통계 조회	`/api/admin/logs/summary/access`	GET	(Query: `date_range`, `group_by=zone|user`)	`Log Service`가 DB에서 집계(Aggregation)
CCTV 영상 URL 조회	`/api/admin/logs/access/{log_id}/video-url`	GET	(없음)	

8. 실시간 통신 (WebFlux SSE 연동)

- **방식:** 서버가 WebFlux의 Server-Sent Events (SSE)를 사용하여 클라이언트로 이벤트를 Push합니다.
- **접속:** Admin UI는 브라우저의 `EventSource` API를 사용하여 `GET /api/admin/events/stream` 엔드포인트에 연결합니다. (Access Token 필요)
- **MIME Type:** `text/event-stream`

목적	이벤트 명 (Event Type)	데이터 형식 (JSON)	비고
실시간 출입 현황	`live_access_event`	`{"user_name": "...", "door_name": "...", "timestamp": "...", "result": "GRANT"}`	`GET /api/admin/events/stream` 구독 시 수신. Kafka의 `access_events` 토픽을 `Log Service`가 WebFlux SSE로 전송.
실시간 장치 상태	`device_status_event`	`{"device_id": "...", "name": "...", "status": "OFFLINE"}`	`GET /api/admin/events/stream` 구독 시 수신. Kafka의 `device_status_events` 토픽을 `Log Service`가 전송.
실시간 알람	`alarm_event`	`{"type": "DOOR_FORCED_OPEN", "door_name": "...", "timestamp": "..."}`	`GET /api/admin/events/stream` 구독 시 수신. Kafka의 `alarm_events` 토픽을 `Log Service`가 전송.

9. 파일 업로드 (MinIO 연동)

- **설명:** 출입자/관리자 사진 업로드를 위한 2단계 프로세스입니다.

목적	엔드포인트	메소드	요청 Body (JSON)	응답 (JSON)	비고
업로드 URL 요청	`/api/admin/uploads/prepare`	POST	`{"filename": "profile.jpg", "content_type": "image/jpeg"}`	`{"upload_url": "https://minio.sdac.local/...", "file_key": "..."}`	`User Service`가 `MinIO`의 Presigned URL을 생성하여 반환
업로드 완료 보고	`/api/admin/uploads/complete`	POST	`{"target_type": "user" | "admin", "target_id": "...", "file_key": "..."}`	(200 OK)	`User Service`가 `users` 또는 `admin_users` 테이블의 `photo_url` 컬럼을 업데이트

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

- **경로:** `/api/admin/commands`

- **설명:** 관리자가 `Admin UI`에서 수동으로 장치를 제어합니다. (높은 권한 필요)

목적	엔드포인트	메소드	요청 Body (JSON)	비고
원격 문 개방	`/api/admin/commands/open-door`	POST	`{"door_id": "...", "reason": "..."}`	`Access Control Service`가 `IP 리더기` / `ACU`에 직접 개방 명령 전송. 감사 로그(Kafka) 필수.
전체 구역 봉쇄	`/api/admin/commands/lockdown`	POST	`{"zone_id": "...", "status": true}`	`Access Control Service`가 `Redis`의 전역 상태를 변경하여 해당 구역의 모든 출입을 차단.
전체 구역 개방	`/api/admin/commands/all-open`	POST	`{"zone_id": "...", "status": true}`	`Access Control Service`가 `Redis`의 전역 상태를 변경하여 해당 구역의 모든 출입을 허용. (화재 시와 유사)

11. 단말기 API (SDAC 버전 Pull 방식)

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

목적	엔드포인트	메소드	요청 Body (JSON)	응답 (JSON)	비고
실시간 출입 판정	`/api/access/attempt`	POST	`{"credential_type": "card", "credential_value": "..."}`	`{"result": "GRANT" | "DENY", "reason": "..."}`	`API Gateway`가 `Device Token` 검증 후 `Access Control Service`로 라우팅. 0.1초 내 응답.
알람/이벤트 보고	`/api/access/event`	POST	`{"event_type": "DOOR_FORCED_OPEN", "timestamp": "..."}`	(200 OK)	도어 센서, 테일게이팅 센서 등의 I/O 컨트롤러가 `API Gateway`로 직접 보고.

12. 비상 API (I/O 컨트롤러 연동)

- **경로:** `/api/emergency`
- **인증:** `X-Device-Token: <secret-key>` 헤더 필요.

목적	엔드포인트	메소드	요청 Body (JSON)	비고
화재 신호 수신	`/api/emergency/fire`	POST	`{"signal_status": "ACTIVE", "source": "FACP-A"}`	모든 정책을 무시하고 `all-open` 실행.

출입 통제 시스템: API 연결 정의서(상세)