DEVELOPER DOCUMENTATION

API documentation

Versioned, token-protected interfaces for domain modules, alerting, control-room integration and license verification.

Updated 13 August 2026 · API v1 · Build.B01.0101

AUTH REQUIRED

Authentication and context

The IGNIS2AB domain and push APIs use a bearer token. Each token is bound to one organisation ID and site ID, displayed in full once, stored only in masked form afterwards and can be revoked at any time.

```
Authorization: Bearer <site-bound-token>
X-Organization-ID: <org-uuid>
X-Site-ID: <site-uuid>
X-Installation-ID: <installation-uuid>  # optional
Content-Type: application/json
```

Token scopes, user permissions, organisation, fire department and object ownership are evaluated together. Never put tokens into URLs, browser code or plaintext logs.

Base URLs

Production
`https://app.ignis2ab.com/api/v1`

Integration

`https://dev.ignis2ab.com/api/v1`

Demo

`https://demo.ignis2ab.com/api/v1`

License service

`https://license.2ab.it/api/v1`

Format

JSON · UTF-8 · ISO 8601 · HTTPS

REST API

Domain API

Organisation- and site-scoped resources use a consistent `data` / `meta` envelope. Collections are paginated.

GET

`/api/v1` – API catalogue

GET / PATCH

`/organization` – tenant and visible fire departments

GET / POST

`/personnel` – list or create site personnel

PATCH

`/personnel/{id}` – update a person in the site

GET

`/vehicles`, `/equipment`, `/incidents`

GET

`/warehouse/articles`, `/training/bookings`, `/finance/earnings-loss`

GET

`/openapi.json` – live API contract

Scopes

```
api:read  api:docs  organization:read
personnel:read  personnel:write  vehicles:read
equipment:read  incidents:read  warehouse:read
training:read  finance:read
```

HTTP PUSH

AlarmDispatcher and control-room API

The push API creates incidents from alerts and processes every change until the mission ends. It accepts event and control-room incident IDs, alert and mission times, keyword, description, location, coordinates, destination, vehicles, call signs, journey numbers, German BOS status and personnel feedback.

GET

`/integrations/alarm-dispatcher/health` – validate token and site

POST

`/integrations/alarm-dispatcher/events` – incident, resources and feedback

POST

`/integrations/alarm-dispatcher/appointments` – appointments, participants and responses

Feedback states

```
accepted | declined | maybe | no_response | unknown
```

Idempotency

A stable `event_id` can be delivered again. Identical content is acknowledged as a duplicate; reusing the ID with changed content returns HTTP 409. Unmatched user IDs are counted and reported.

CONNECT API

Vendor-neutral alert integration

Other providers can create, update, close or end alerts and missions and submit feedback and resources independently.

```
GET    /api/v1/connect/health
POST   /api/v1/connect/alarms
PATCH  /api/v1/connect/alarms/{externalId}
POST   /api/v1/connect/alarms/{externalId}/feedback
POST   /api/v1/connect/alarms/{externalId}/resources
POST   /api/v1/connect/alarms/{externalId}/close
POST   /api/v1/connect/missions
PATCH  /api/v1/connect/missions/{externalId}
```

```
DELETE /api/v1/connect/missions/{externalId}
```

Example: mission update

```
POST /api/v1/integrations/alarm-dispatcher/events
{
  "event_id": "adc-event-20260813-001",
  "event_type": "mission.updated",
  "occurred_at": "2026-08-13T18:31:22+02:00",
  "mission": {
    "id": "adc-mission-4711",
    "control_center_incident_number": "LS-MSH-2026-004711",
    "status": "active",
    "keyword": "B2 Building",
    "alarmed_at": "2026-08-13T18:27:03+02:00",
    "location": {"street":"Example
Street","house_number":"12","city":"Allstedt"}
  },
  "vehicles": [{"id":"40-11-1","radio_call_sign":"Florian 40/11-
1","status":"3"}],
  "personnel": [{"user_id":"adc-user-
88421","status":"accepted"}]
}
```

Success responses include the IGNIS incident ID and counts for resources, feedback and unmatched responses.

2AB License API

The independent license service verifies tenant and site licenses, binds organisation, site and installation UUIDs and returns mode, validity, features and limits. Verification uses client ID, timestamp, nonce and an HMAC-SHA256 signature. Controlled trial provisioning uses a server-side bearer token.

POST

`https://license.2ab.it/api/v1/licenses/verify`

POST

`https://license.2ab.it/api/v1/licenses/trials`

Status codes and operation

200 / 202	
Processed or idempotent duplicate	
400	
Invalid JSON	
401	
Missing, invalid or expired token	
403	
Scope, organisation/site assignment or integration forbidden	
409	
Idempotency conflict	
415	
Content type is not JSON	
422	
Missing or invalid field	

Keep secrets separate per environment and rotate them regularly. Production integrations should use retries with backoff, stable external IDs and their own delivery

log.