# RBA-01 | RBAC & RLS

User Stories

# Access Control & Security Management

## Overview

**Three Layer Configuration:**

- RBAC Role (Admin & User) Defines general privilege tier, who can configure vs. who can use.
- Feature Access - Defines what modules the users can access and see.
- Access Control (RLS) Defines what data they can view within the modules and the map navigation

## Roles & Core Permissions

| Role | Description | Core Capabilities |
|---|---|---|
| **Admin** | Full access across all modules and datasets. Manages users, audits, and data restrictions. | Full feature + data access |
| **User** | Limited, Enforced by access rules | Restricted modules, datasets, location and redacted outputs. |

## RBAC Feature-Level Access Matrix

Controls which modules and features each client can access:

For Users, feature-level access is not static, it’s customizable per individual.

Admins can selectively enable any combination of modules.

| Module | Description |
|---|---|

| | |
|---|---|
| **Monitor Mode** | Playbacks, Event Viewing & Data Feed |
| **Motion Tracer** | Device Tracer & Route Analysis |
| **Network Graph** | Device-to-ip and association mapping |
| **AI Assistant** | Live & Historical Queries or Data & Analysis |
| **Profile Device Data** | Device Metadata and Contextual Info |
| **Audit Logs** | Access to users activity history for performance audit |
| **Data Export** | Download/Reports/Map Shots export |

# RLS Customizable Data Access (Per User)

| Field | Type | Description |
|---|---|---|
| user_id | UUID | Supabase Auth ID |
| modules_enabled | TEXT[] | e.g. ['monitor','tracer'] |
| region_geom | GEOGRAPHY | Geographic boundary polygon |
| record_limit | INT | Max rows retrievable |
| time_window_days | INT | Lookback window (e.g. 30) |
| mask_shodan | BOOLEAN | Hide Shodan fields if TRUE |
| hash_identifiers | BOOLEAN | Hash device_id & IPs if TRUE |
| ai_assistant_access | BOOLEAN | Enable/disable AI assistant |
| created_at | TIMESTAMP | Audit timestamp |

## RLS Layer — Data-Level Security

Implemented at Supabase level to restrict data access based on:

- **Geographic boundaries** (city, state, region) using PostGIS
- **Volume limits** (record count caps for trial users)
- **Temporal ranges** (time-based data access windows)

## Client Access Scenarios

### Client A — Feature-Restricted Access

- **Access**: Profile Modal only
- **Restrictions**: No Tracer or Connections modules
- **Use Case**: Basic device lookup without behavioral analysis

**Client B — Geographic Restriction**

- **Access**: Full features (Profile, Tracer, Connections, AI Panel)
- **RLS Filter**: Miami geographic boundary only
- **Implementation**: Supabase RLS policy filters data by lat/lon bounds
- **Use Case**: Regional operations with location-specific intelligence

**Client C — Volume-Limited Trial**

- **Access**: Full features
- **RLS Filter**: Maximum 10,000 records per query
- **Implementation**: Supabase RLS policy with LIMIT clause
- **Use Case**: Trial period with controlled data exposure

## Implementation Requirements

### 1. Role and Access Configuration

- Maintain two core roles (Admin, User) managed through Supabase Auth.
- Store all per-user configurations (modules, geography, record limits, temporal scope, masking, hashing) in `user_access_rules`.
- Admins can dynamically assign or revoke access via configuration UI or database updates.
- Include version tracking for access-rule changes (to support rollback and audit history).

---

### 2. Supabase RLS Policies

- **Geographic boundary** enforcement using PostGIS `ST_Within()` for spatial filtering.
- **Temporal access** limits via timestamp-based range filters.
- **Record limits** applied through query-level `LIMIT` enforcement or function-based result caps.
- **Identifier hashing and masking** (IPs & Device Ids)
- **AI Assistant RLS Integration:** All assistant queries route through RLS-filtered (`arkem_filtered_context`).
- Policies tested against each module's expected query patterns to ensure consistent restriction logic.

---

### 3. Audit Logging (This will be as we build an audit system but the initial "set up" can be there)

- All access and actions logged in `arkem_audit_logs` and `user_access_rules_log`.
- Each log entry includes: `user_id`, `module`, `action`, `records_returned`, `sensitivity_tier`, `timestamp`, and `ip_address`.
- Logs separated by access type:
  - **System Actions:** Configuration changes, exports, failed logins.
  - **Data Actions:** View, query, export, AI query.
- Include retention policy (e.g. 12 months default) and export options for compliance or investigation reports.
- (*Optional*): Real-time log dashboard and anomaly alerts (e.g. unusual volume or repeated failed queries).

---

### 4. Security Standards

- JWT-based authentication with embedded claims: `role`, `user_id`, and `tier`.
- All endpoints and data transmissions secured via **TLS 1.3**.

- Sensitive fields (device IDs, IPs) **hashed for non-admins**; raw identifiers accessible to Admins only.
- Implement **row-level masking** for Shodan enrichment and critical metadata.
- Regular **penetration testing and RLS validation checks**.
- Compliance alignment: **GDPR**, **SOC 2**, **CJIS** (if future law enforcement partners).
- (*Optional*): Automated session expiration and token refresh lifecycle.

---

### 5. Data Isolation & Multi-Tenancy Readiness

- All user data filtered by `client_id` when multi-tenant mode is introduced.
- Ensure cross-client data separation at both RLS and API layers.
- (*Future consideration*): Dedicated schema per client for high-sensitivity operations.

---

### 6. Feature Integration Dependencies

- Frontend middleware validates `modules_enabled` array before loading routes or components.
- Each module's data API validates the same configuration server-side (no UI-only enforcement).
- Shared enforcement service (middleware or Supabase function) to centralize rule checking across all modules.

*Possible Consideration is the for Admin to Create Predefined Roles (AFTER MVP)