L I C E L I T E C H N O L O G I E S

Direction Technique · Plateforme SMART Audit

S P É C I F I C A T I O N T E C H N I Q U E

Gestion du licensing

Cas SMART Audit — Banques UEMOA

Modèle retenu

Activation en ligne, fonctionnement hors-ligne

Référence	LICELI-SPEC-LICENSING-AUDIT
Version	1.0
Date	Mai 2026
Périmètre	Plateforme SMART Audit (déploiement on-premise UEMOA)
Statut	Document de référence interne
Confidentialité	Interne LICELI

Table des matières

> **Modèle retenu :** *activation en ligne, fonctionnement hors-ligne*, avec un **numéro de licence court** côté client (esprit Odoo) et un **jeton signé** côté technique pour la vérification sans réseau.

Légende des marqueurs : ✅ *déjà en place* · 🟡 *partiel* · 🔧 *cible (à implémenter).*

Sources : `Licensing-Hub/backend/{catalog,licensing}` et `audit_backend_v2/licensing` (branche `feature/licensing`).

1. Objet et périmètre

Définir comment une licence du produit **SMART Audit** est émise par le **Hub** (serveur central LICELI qui émet et gère les licences), activée chez le client, puis appliquée par l'application pour ouvrir ou fermer ses fonctionnalités.

Le document couvre : le format des artefacts (numéro court et jeton signé), le cycle de vie (émission, activation, fonctionnement, renouvellement, downgrade, révocation), l'application des droits côté Audit, la gestion des clés, le modèle de menaces, et les paramètres recommandés.

2. Vocabulaire (termes définis)

Terme	Définition
Hub	serveur central LICELI qui émet et gère les licences (« Liceli Core »).
Produit consommateur	l'application qui vérifie la licence (ici SMART Audit).
Cryptographie asymétrique	système à deux clés, une privée (signe) et une publique (vérifie). Ce qui est signé par la privée n'est vérifiable que par la publique correspondante.
Ed25519	algorithme de signature numérique (RFC 8032), rapide et sûr.
Jeton signé	chaîne base64url(charge_utile).base64url(signature) portant les droits, vérifiable hors-ligne.
Numéro de licence	référence courte, lisible, communiquée au client (ex. AUDIT-2026-A4B7-9F2E).
Module	brique fonctionnelle activable (unité de licence). Pour Audit : M1 / M2 / M3.
Plan	bundle commercial de modules avec prix.
Activation	enregistrement, côté Hub, d'un déploiement réel utilisant la licence.
Identifiant d'installation	nombre aléatoire unique généré une fois côté client et stocké dans le volume de données ; identité stable de l'installation.
Empreinte matérielle	condensé des caractéristiques physiques du serveur (machine-id, MAC, disque, hostname). Fragile, abandonnée comme identité (voir §9).
HTTP 402	code « Payment Required », renvoyé quand un module n'est pas couvert.
CRL (Certificate Revocation List)	liste signée des licences révoquées.

Terme	Définition
Ratchet d'horloge	mécanisme qui mémorise (de façon signée) le dernier instant vu et détecte un recul de l'horloge système.
Période de grâce	délai pendant lequel un module retiré reste utilisable avant bascule en lecture seule.

3. Vue d'ensemble

3.1. Deux axes de compartimentage

1. **Axe commercial (catalogue, Hub)** — *ce qui est vendable :* `Product → Module → Plan`.
2. **Axe d'exécution (émetteur ↔ consommateur)** — *qui a le droit, où :* le Hub émet une licence signée ; SMART Audit la vérifie et gate ses fonctionnalités module par module.

3.2. Modèle de déploiement retenu : activation en ligne, fonctionnement hors-ligne

- Le client ouvre temporairement l'accès réseau sortant vers le Hub le temps d'activer (et de renouveler), puis le referme.
- Entre deux activations, l'application fonctionne sans réseau : elle vérifie le jeton signé en local.
- Le client ne manipule qu'un numéro de licence court ; le jeton signé est échangé et stocké automatiquement.

Figure 3 — Cloisonnement cryptographique : Hub LICELI ⇄ SMART Audit

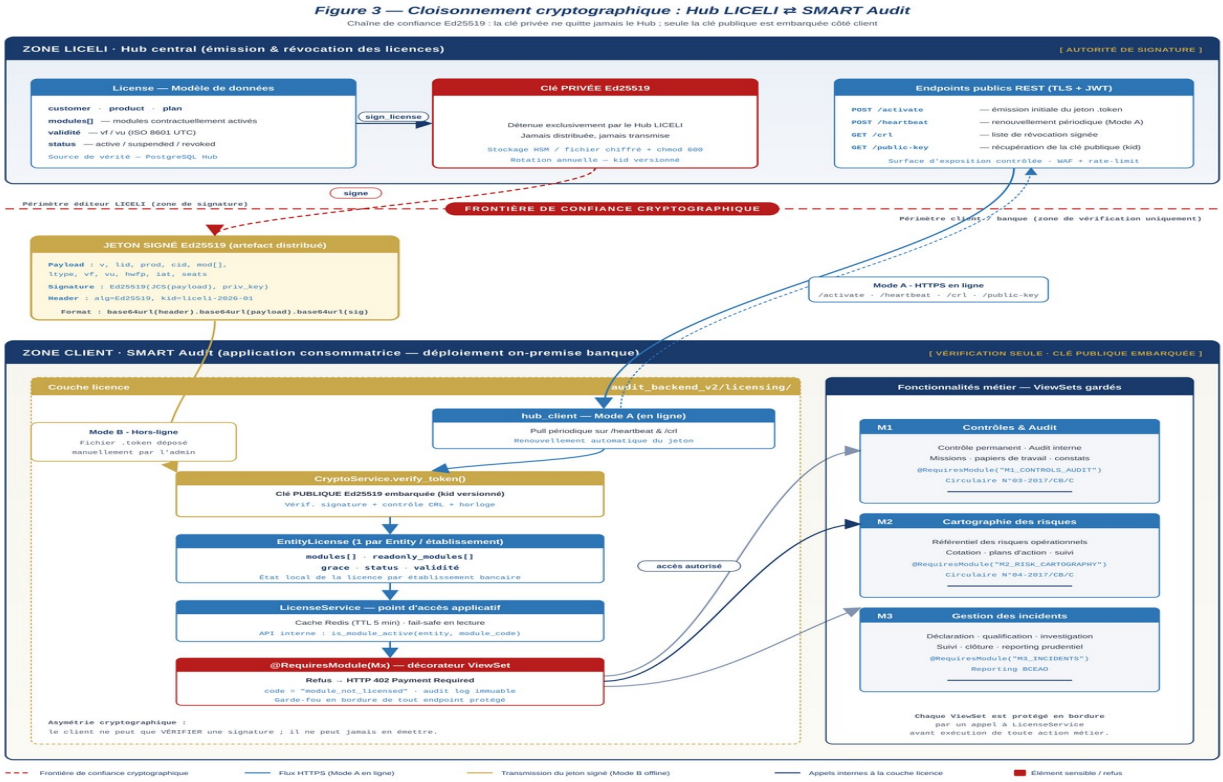

Figure 1 — Cloisonnement cryptographique Hub LICELI ⇄ SMART Audit

4. Acteurs et composants

Composant	Rôle	Clé détenue
Hub (Django)	émet, signe, révoque, suit les activations	**clé privée** (signe)
SMART Audit backend (Django)	vérifie le jeton, applique les droits (`RequiresModule`), stocke `EntityLicense`	**clé publique** (vérifie)
SMART Audit frontend (Next.js)	affichage uniquement : cadenas, redirection sur 402	aucune (jamais de sécurité côté navigateur)

> **Règle : toute la sécurité est côté backend.** Le frontend ne fait que réagir aux réponses du serveur.

5. Modèle de données

5.1. Côté Hub (catalog + licensing) — ✅

- **Product** : `code` (repris dans le jeton `prod`), `deployment_type` (saas / paas / iaas / baas / onprem).
- **Module** : `product`, `code` (ex. `M1_CONTROLS_AUDIT`), `order`.
- **Plan** : `product`, `modules` (M2M), `billing_period`, `price`, `currency` (défaut XOF), `seats`, `downgrade_grace_days` ✅ (cf. §13).
- **Customer** : `uuid`, `code`, `name` — l'organisation cliente.
- **License** : `customer`, `product`, `plan`, `license_number`, `modules[]`, `readonly_modules[]`, `license_type`, `status` (pending / active / suspended / expired / revoked), `valid_from` / `valid_until`, `hardware_fingerprint`, `seats`, `max_activations`, `downgrade_grace_days` ✅, `signed_token`, `signed_at`.
- **Activation** : `license`, `fingerprint`, `hostname`, `ip_address`, `last_heartbeat`, `is_active`.
- **Subscription** : cycle commercial (trial / active / past_due / canceled), `entitlements()` au runtime (pour les produits SaaS sans clé).
- **RevocationEntry** → alimente la **CRL** signée.

5.2. Côté SMART Audit (licensing) — ✅

EntityLicense (par `Entity`) : `license_number`, `modules[]`, `readonly_modules[]`, `module_grace_until{}`, `downgrade_grace_days`, `license_type`, `status` (pending_activation / active / grace / suspended / revoked / expired), `valid_from` / `valid_until`, `signed_token`, `hardware_fingerprint`.

> **Contrainte d'intégrité :** *une seule licence active par Entity.*

6. Modules SMART Audit — ✅

Module	Couvre	Dépend de
`M1_CONTROLS_AUDIT`	Contrôles + audit interne (contrôles niveau ≥ 3)	— (socle)
`M2_RISK_CARTOGRAPHY`	Cartographie des risques	`M1`
`M3_INCIDENTS`	Gestion des incidents (`ExternalIncident`)	`M1`

Bundles valides : `[M1]`, `[M1, M2]`, `[M1, M2, M3]`. M1 est obligatoire ; les bundles sans M1 sont refusés à la validation.

7. Le numéro de licence (artefact client) — ✅

Référence courte, lisible, communiquée au client (l'équivalent du code d'abonnement Odoo).

- **Format** : `<PRODUIT>-<ANNÉE>-XXXX-XXXX` (XXXX = 4 caractères hexadécimaux aléatoires).
- **Exemple** : `AUDIT-2026-A4B7-9F2E`.
- **Génération** : `generate_license_number(product_code)` (Hub), aléatoire cryptographique, unicité garantie en base.
- **Usage** : c'est ce que le client saisit ou communique pour activer. Il ne voit pas le jeton long.

> 🔧 **À fixer :** *préfixe par produit (AUDIT, MOBILE, …), casse, éventuelle clé de contrôle (checksum) pour détecter les fautes de frappe.*

8. Le jeton signé (artefact technique) — ✅

La licence réelle, vérifiable hors-ligne.

- **Format** : `base64url(charge_utile_json_canonique) + "." + base64url(signature_Ed25519)` (structure proche d'un JWT, mais 2 parties).

Charge utile (payload) :

```
{
  "v": 1,
  "lid": "AUDIT-2026-A4B7-9F2E",        // numéro de licence
  "prod": "SMART_AUDIT",                 // code produit
  "cid": "<identifiant client/entity>",
  "mod": ["M1_CONTROLS_AUDIT", "M2_RISK_CARTOGRAPHY"],  // modules write
  "ro": [],                              // modules en lecture seule
  "ltype": "onprem",                     // onprem | saas
  "vf": "2026-01-01T00:00:00Z",          // début de validité
  "vu": "2027-01-01T00:00:00Z",          // fin de validité
  "hwfp": "",                            // liaison machine (vide = non lié) — voir
§9
  "iat": "2026-01-01T00:00:00Z",         // émis le
  "seats": 25,                           // sièges
```

```
  "grace": 30                              // jours de grâce au downgrade (§13)
}
```

- **Signature** : Ed25519 sur le JSON canonique (clés triées, séparateurs compacts).
- **Vérification** : `CryptoService.verify_token` (Audit backend) avec la **clé publique**. Toute altération d'un seul octet invalide la signature.

Figure 2 — Payload type de licence (Licensing Engine Ed25519)

Structure du token signé : `base64url(header) . base64url(payload) . base64url(Ed25519_signature)`

PAYLOAD · application/json

```
{
  "v": 1,
  "lid": "AUDIT:2026:A4B7:9F2E",
  "prod": "SMART_AUDIT",
  "cid": "550e8400-e29b-41d4-a716-446655440000",
  "mod": ["M1_CONTROLS_AUDIT", "M2_RISK_CARTOGRAPHY"],
  "ltype": "onprem",
  "vf": "2026-05-21T00:00:00Z",
  "vu": "2027-05-21T00:00:00Z",
  "hwfp": "sha256:abc123def4567890...",
  "iat": "2026-05-21T10:00:00Z",
  "seats": 1
}

// Bloc de signature détaché (transporté avec le payload)
{
  "alg": "Ed25519",
  "kid": "liceli-licensing-2026-01",
  "sig": "3045a2c1...e8f4b9d6"  // 64 octets / base64url
}

Encodage canonique : JSON RFC 8785 (JCS) avant signature
Vérification : Ed25519.verify(public_key[kid], JCS(payload), sig)
```

SPÉCIFICATION DES CHAMPS — v = 1

CHAMP	TYPE	DESCRIPTION	REQUIS
`v`	int	Version du schéma de payload (entier, actuellement 1).	OUI
`lid`	string	License ID — identifiant unique de la licence. Format : `PROD:YEAR:HEX4:HEX4`	OUI
`prod`	enum	Code produit ciblé. `SMART_AUDIT | BADF_MOBILE | PI_API_BUSINESS | …`	OUI
`cid`	uuid	Customer ID — identifiant universel du client (banque). UUID v4 — référentiel central du Hub	OUI
`mod`	array	Liste des modules activés (codes contractuels). Sous-ensemble du catalogue produit, non vide	OUI
`ltype`	enum	License type — modèle de déploiement. `onprem | saas`	OUI
`vf`	datetime	*Valid From* — début de validité (ISO 8601 UTC).	OUI
`vu`	datetime	*Valid Until* — expiration de la licence (ISO 8601 UTC).	OUI
`hwfp`	string	*Hardware fingerprint* — verrou matériel SHA-256. Requis si `ltype = onprem` · `null` sinon	COND.
`iat`	datetime	*Issued At* — horodatage d'émission de la licence.	OUI
`seats`	int	Nombre d'instances ou postes utilisateurs autorisés. Entier > 0 · contrôlé côté Service d'autorisation	OUI

OUI Champ obligatoire — COND. Conditionnel (selon ltype)

Règles de validation côté Service d'autorisation

1. Vérification de la signature Ed25519 contre la clé publique identifiée par **`kid`** (rotation annuelle).
2. Contrôle temporel strict : **`vf ≤ now < vu`** ; rejet si la licence est révoquée (consultation cache Redis + CRL signée).
3. Pour **`ltype = onprem`** : vérification que **`hwfp`** correspond à l'empreinte matérielle calculée à l'exécution ; pour **`ltype = saas`** : isolation par **`cid`** et compteur de tenants ≤ **`seats`**.

Figure 2 — Payload type de licence et spécification des champs

9. Identité d'installation (remplace l'empreinte matérielle) — 🔧 CIBLE

Problème de l'empreinte matérielle *(état actuel ✅ mais fragile)* : condensé de machine-id + MAC + UUID disque + hostname + sel. Elle **change** au moindre changement d'environnement (reconstruction de conteneur, migration, changement de carte réseau) → faux blocages et ré-activations subies.

Cible : lier la licence à un **identifiant d'installation** :

- nombre aléatoire unique généré à la première activation ;
- stocké dans le **volume de données persistant** (survit aux redémarrages, reconstructions d'image, migrations si le volume suit) ;
- envoyé au Hub à chaque (ré-)activation → c'est l'identité que le Hub voit (même principe que le `database.uuid` d'Odoo).

Conséquence : pendant le fonctionnement hors-ligne, l'application ne revérifie pas le matériel → un

> changement d'environnement ne bloque plus rien. La ré-activation n'a lieu qu'au renouvellement (événement planifié).

> 🔧 **À implémenter :** *génération + persistance de l'identifiant ; bascule du binding* `hwfp` *(matériel) →* `install_id` *(logiciel) dans le jeton et la vérification.*

10. Cycle de vie

10.1. Émission (Hub) — ✅

`issue_license` : choisit modules + plan + durée → crée la `License` → la **signe** (clé privée) → produit le `signed_token`. Statut `pending` → `active`.

10.2. Activation en ligne — ✅ (endpoint) / 🔧 (identité)

Séquence d'activation :

```
1.  Client (ouverture des ports sortants)
              ↓
2.  POST /activate/  { license_number, install_id, hostname }
              ↓
3.  Hub  →  vérifie max_activations (anti sur-déploiement)
              ↓
4.  Hub  →  { signed_token, modules, valid_until }
              ↓
5.  Audit  →  verify_token (clé publique)  →  EntityLicense (status=active)
              ↓
6.  Client (fermeture des ports)
```

10.3. Fonctionnement hors-ligne — ✅

À chaque requête (avec cache 5 min), `LicenseService.get_active_license` exige `status == active` **ET** `valid_from ≤ maintenant ≤ valid_until`. La vérification de signature (mode strict) lie le jeton à la base (anti-falsification en base).

10.4. Renouvellement — ✅ (mécanisme) / 🔧 (auto-rebind)

À l'approche de `valid_until` : ré-ouverture ponctuelle des ports → ré-activation → nouveau jeton avec une `valid_until` repoussée. **Pas de reconstruction d'image.**

> 🔧 **Cible :** *re-liaison automatique. Si l'identité a changé mais reste dans* `max_activations`*, le Hub re-lie sans intervention.*

10.5. Downgrade / grâce / lecture seule — ✅

Module retiré → période de **grâce** (`downgrade_grace_days`, encore pleinement actif) → à l'expiration, bascule en **lecture seule** (`readonly_modules`) : consultation oui, écriture → HTTP 402. **Les données ne sont jamais supprimées.**

10.6. Révocation — ✅ / ⚠️ hors-ligne

`RevocationEntry` → **CRL** signée. Prise en compte à la synchro (en ligne) ou au prochain renouvellement.

> ⚠️ **Limite :** *un site hors-ligne n'est pas révocable en temps réel (borné par la durée de validité).*

Figure 4 — Cycle de vie d'une licence SMART Audit et de ses modules

Conforme à SPEC-LICENSING-AUDIT.md — Deux machines à états : niveau licence (haut) et niveau module au sein d'une licence active (bas)

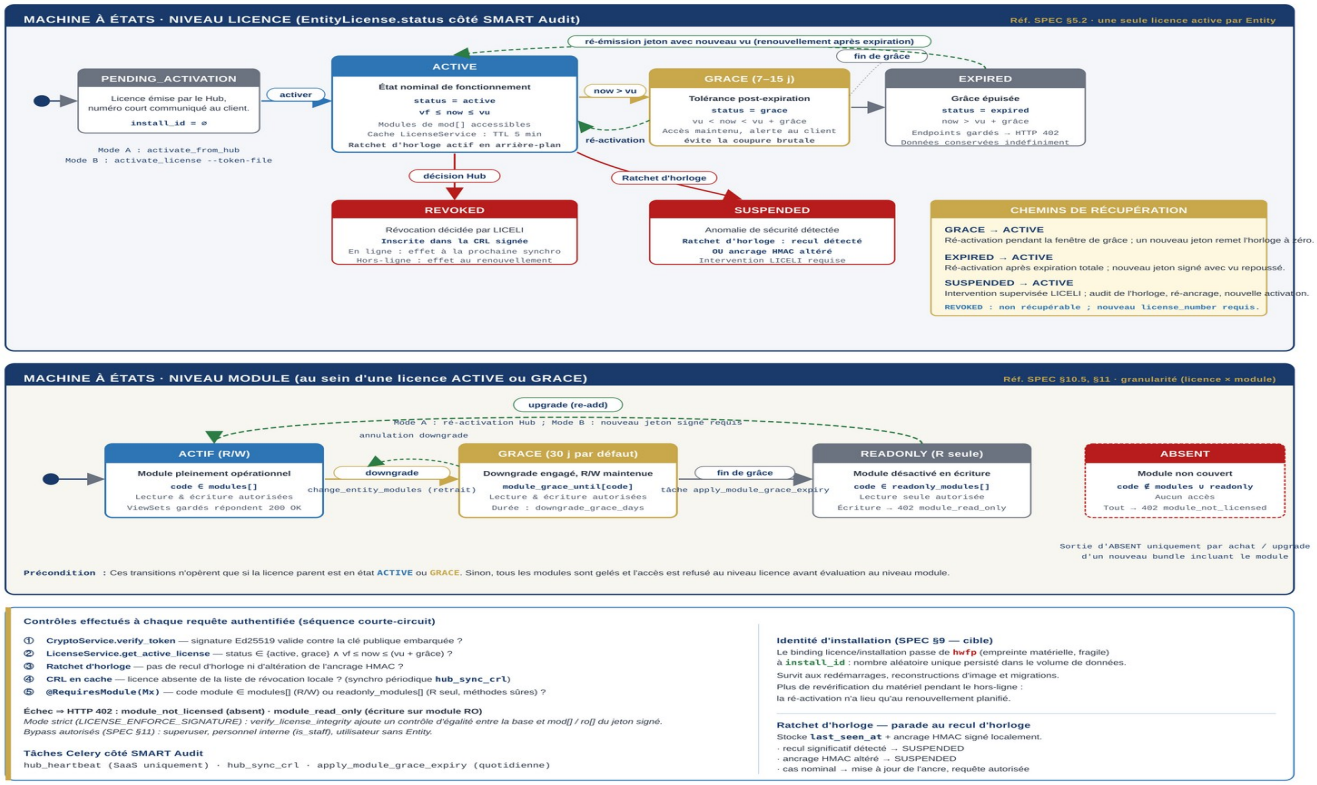

Figure 3 — Cycle de vie d'une licence et de ses modules

11. Application des droits côté Audit — ✅

- Permission DRF `RequiresModule(Mx)` sur les ViewSets :
  - module absent → **HTTP 402** `module_not_licensed` ;
  - écriture sur un module en lecture seule → **HTTP 402** `module_read_only` ;
  - méthodes sûres (GET) autorisées si le module est lisible (actif, grâce ou lecture seule).
- **Bypass** : superuser, personnel interne (`is_staff`), utilisateur sans Entity.
- **Mapping** : M1 → Contrôles / Audit, M2 → Cartographie, M3 → Incidents.
- **Frontend** : intercepte le 402 → page « fonctionnalité non incluse » + cadenas. Aucune décision d'accès côté navigateur.

12. Gestion des clés — ✅

Clé	Rôle	Emplacement	Secrète
Privée (`LICELI_LICENSE_P`	signer	**uniquement le Hub**, via variable d'environnement / coffre à secrets (KMS / HSM)	🔴 **oui**

Clé	Rôle	Emplacement	Secrète
`RIVATE_KEY)`		— jamais dans git ni dans une image	
Publique (`PUBLIC_KEY_HEX`)	vérifier	**dans le backend SMART Audit** (constante ou variable d'env)	🟢 **non**

- La clé privée **ne quitte jamais le Hub** et n'est jamais livrée à un client.
- La clé publique dans l'app n'est pas un risque de fuite (elle ne fait que vérifier).
- 🔧 Prévoir une **procédure de rotation** (changer la paire + diffuser la nouvelle clé publique lors d'une mise à jour produit).

13. Délai de grâce configurable depuis le Hub — ✅ (Hub) / 🔧 (Audit)

- ✅ `Plan.downgrade_grace_days` (défaut configurable par plan) → hérité par `License.downgrade_grace_days` à l'émission (surchargeable) → embarqué dans le jeton (`grace`).
- ✅ `issue_license` (service + commande `--grace-days` + API) honore la valeur.
- 🔧 Côté Audit : lire `payload["grace"]` à l'activation pour alimenter `EntityLicense.downgrade_grace_days`, et le **lier** dans la vérification d'intégrité (anti-modification en base).

14. Modèle de menaces et parades

#	Menace	Gravité	Parade
1	**Copie du volume** pendant le hors-ligne (clone qui tourne jusqu'à expiration)	**élevée**	durée de validité courte + détection de doublon au renouvellement + max_activations
2	**Recul d'horloge** / restauration d'un instantané pour ne jamais expirer	**moyenne**	expiration sur la date signée + ratchet d'horloge + validité courte
3	**Client administrateur (root)** : patch du code ou remplacement de la clé publique	**élevée / peu probable (banques)**	intégrité au démarrage + composant critique compilé + contrat / audit (imbattable purement techniquement)
4	**Sur-déploiement** (même numéro sur N installations)	**moyenne**	max_activations strict à l'activation (fiable car activation en ligne)
5	**Fuite de la clé privée du Hub**	**critique**	coffre à secrets / HSM, accès restreint, rotation
6	**Révocation impossible hors-ligne**	**moyenne**	validité courte → effet au prochain renouvellement
7	**Interception réseau à l'activation** (MITM)	**faible**	HTTPS / TLS + vérification de signature + épinglage de l'URL Hub

> **Vérité de fond :** on-premise + client root = pas d'anti-copie « dur ». Objectif réaliste : empêcher l'abus accidentel, le rendre **coûteux et détectable au renouvellement**, et s'appuyer sur le contrat.

Trois leviers couvrent l'essentiel : **(1) validité courte, (2) max_activations strict, (3) clé privée protégée.**

15. Paramètres recommandés — 🔧 à valider

Paramètre	Recommandation	Compromis
Durée de validité du jeton	**6 à 12 mois**	plus court = plus de contrôle, ré-ouverture des ports plus fréquente
Délai de grâce après expiration	**7 à 15 jours**	évite une coupure brutale en cas de retard d'ouverture des ports
`max_activations` par licence	**1 (+1 ou 2 pour migration / secours)**	trop haut = sur-déploiement facile
`downgrade_grace_days`	**30 jours (par plan)**	temps d'exporter / régulariser avant lecture seule
Fréquence du heartbeat (si connecté)	**horaire**	optionnel ; surtout utile en SaaS

16. Interfaces

Hub — endpoints publics (/api/v1/licensing/) ✅

- `POST /activate/` — active une licence (renvoie le jeton).
- `POST /heartbeat/` — preuve de vie (SaaS / clients connectés).
- `GET /crl/` — CRL signée.
- `GET /public-key/` — clé publique (hex).

Hub — commandes ✅

- `issue_license --customer … --product … --modules … --days … [--grace-days …]`
- `change_entity_modules` (upgrade / downgrade) — côté Audit.

SMART Audit ✅

- `activate_from_hub` (Mode A en ligne), `activate_license --token-file` (Mode B offline).
- Tâches : `hub_heartbeat`, `hub_sync_crl`, `apply_module_grace_expiry`.

17. État actuel vs cible (récapitulatif)

Élément	État
Jeton signé Ed25519, numéro court, vérification offline	✅

Élément	État
Modules M1 / M2 / M3, `RequiresModule` → 402, grâce / lecture seule	✅
Grâce configurable depuis le Hub (Plan / License / jeton)	✅ Hub · 🔧 lecture côté Audit
Activation en ligne (endpoint)	✅
Identifiant d'installation (remplace l'empreinte matérielle)	🔧
`max_activations` strict + détection de doublon au Hub	🟡 (champ présent, contrôle à renforcer)
Re-liaison automatique au renouvellement	🔧
Procédure de rotation de clé + coffre à secrets en production	🔧
Paramètres (validité, grâce) figés	🔧 à valider

18. Décisions ouvertes

3. Durée de validité et délai de grâce définitifs (§15).
4. Format final du numéro de licence (§7).
5. Implémenter l'identifiant d'installation et basculer le binding (§9).
6. Renforcer le contrôle `max_activations` + détection de doublon côté Hub (§14 #1 / #4).
7. Procédure de stockage / rotation de la clé privée en production (§12).

Annexe — Documents de référence

- `ARCHITECTURE-COMPARTIMENTAGE-LICENCE.md` — schémas d'architecture détaillés.
- `img/audit-licence.png` — diagramme historique source.
- `ROADMAP-ETAPE7-SAAS-PAIEMENTS.md` — intégration paiements et automatisation upgrade / downgrade.

> **Fin du document.** *Ce document est interne à LICELI Technologies. Les éléments marqués 🔧 constituent la feuille de route technique de l'équipe SMART Audit.*