

Authentification Pronote

Référence technique pour pronote-sync

Version 1.0 — 12 septembre 2026

Équipe Architecture et Sécurité

Ce document n'est pas un guide officiel. Il n'est ni approuvé, ni validé, ni autorisé par le Ministère de l'Éducation Nationale française ni par Docaposte (éditeur de Pronote / Index Éducation).

AVERTISSEMENT

Ce document repose sur des observations directes du code source de pronotepy (version 2.15.7, vérifiée localement le 2026-09-12), des issues publiques du dépôt bain3/pronotepy, et des tests d'intégration du projet pronote-sync.

Un essai réel sur l'instance Pronote cible est nécessaire avant tout usage en production. Les tests/mocks ne prouvent pas la compatibilité d'instance.

Table des matières

Avertissement initial	3
1 Introduction	4
1.1 Périmètre	4
1.2 Public cible	4
2 Tableau de compatibilité avec pronote-sync	5
3 Méthode 1 : URL iCal sécurisée (icalsecurise)	6
3.1 Principe général	6
3.2 Obtention du jeton	6
3.3 Format de l'URL	6
3.4 Cycle de vie	7
3.5 Sécurité	7
3.6 Intégration dans pronote-sync	8
4 Méthode 2 : Connexion directe (identifiant / mot de passe + ENT)	9
4.1 Principe général	9
4.2 Flux d'authentification	9
4.3 Intégration dans pronote-sync	10
4.4 ENT supportés	10
5 Méthode 3 : QR Code + token mobile (pour pronote-sync)	11
5.1 Principe général	11
5.2 Procédure QR pour pronote-sync	11
5.2.1 Étape 1 : Génération du QR code (interface web)	11
5.2.2 Étape 2 : Configuration de pronote-sync	12
5.2.3 Étape 3 : Premier login (enrôlement)	12
5.2.4 Étape 4 : Connexions ultérieures (auto-login)	13
5.3 Cycle de vie des tokens	13
5.4 Sécurité	14
5.5 Erreurs et repli	14
5.6 Incompatibilités	15
6 Méthode 4 : SSO ENT (CAS/SAML)	16
7 Méthode 5 : EduConnect (HubEduConnect)	17
8 Méthode 6 : CAS direct	18

9	Méthode 7 : API publique	19
A	Bibliothèques tierces	20
B	Tableau comparatif synthétique	21
	Légende	22
C	Écarts et notes de cohérence	23
	C.1 Alignement avec <code>.env.example</code>	23
	C.2 Écarts avec <code>docs/exploitation.md</code>	23
	Glossaire	24
	Historique des révisions	25

Liste des tableaux

2.1	Méthodes d'authentification et leur statut dans pronote-sync	5
A.1	Bibliothèques tierces et leur statut dans pronote-sync	20
B.1	Comparatif des méthodes d'authentification	21
C.1	Alignement avec .env.example	23
C.2	Écarts avec docs/exploitation.md	23

Avertissement initial

Convention de qualification :

-  **Observé dans le code local** : Mécanisme vérifié dans le code source de pronote.py 2.15.7 ou pronote-sync.
-  **Observé localement** : Comportement constaté dans les interfaces Pronote (version non spécifiée, hypothèse à valider).
-  **Hypothèse à valider** : Affirmation non vérifiée, nécessitant une confirmation par test sur une instance réelle.

Note importante (2026-09-12) :

Un essai réel sur l'instance Pronote cible est nécessaire avant tout usage en production. Les tests/mocks ne prouvent pas la compatibilité d'instance.

Chapitre 1

Introduction

Ce document documente **uniquement les méthodes d'authentification prises en charge ou déléguables par pronote-sync**, en alignement strict avec :

- Le code source de `pronote_sync/sources/pronote/client.py` (liste fermée `_ENT_NAMES`, gestion des tokens).
- La bibliothèque `pronotepy 2.15.7` (contrat des méthodes `qrcode_login`, `token_login`, `export_credentials`).
- Les paramètres de configuration `.env.example` (lignes 21-25).

1.1 Périmètre

- ✓ **Pris en charge** : Méthodes implémentées et testées dans `pronote-sync`.
- ✎ **Déléguable à pronotepy** : Méthodes gérées par `pronotepy` mais non directement exposées par `pronote-sync`.
- ✗ **Non implémenté** : Méthodes non supportées (ex. CAS/SAML générique, EduConnect direct).

1.2 Public cible

Développeurs et intégrateurs de `pronote-sync`.

Chapitre 2

Tableau de compatibilité avec pronote-sync

Table 2.1 : Méthodes d'authentification et leur statut dans pronote-sync

Méthode	Statut dans pronote-sync	Implémentation	Source	Notes
URL iCal sécurisée (icalsecurise)	✓ Pris en charge	Source ical (prioritaire en mode auto)	 pronote_sync/sources/ical.py	Jeton dans l'URL traité comme secret.
Connexion directe (mot de passe + ENT)	✓ Pris en charge	Source pronotepy (repli si iCal échoue)	 pronote_sync/sources/ent.py	Liste fermée ENT-NAME (lignes 52-84).
QR Code + token mobile	✓ Pris en charge	Mode PRONOTE_AUTH_MODE_PONOTEPY	 pronotepy.ParentClient.login + token_login	Voir Procédure QR (Section code).
SSO ENT (CAS/SAML)	✗ Non implémenté	—	—	Seuls les 30 ENT de la liste fermée _ENT_NAMES (pronotepy 2.15.7) sont supportés via ent dans ParentClient. Pas de SSO générique CAS/SAML.
EduConnect (HubEduConnect)	✗ Non implémenté	—	—	Nécessite une intégration CAS/SAML générique, non supportée par ce projet.
CAS direct	✗ Non implémenté	—	—	Non supporté.
API publique	✗ Inexistante	—	—	Aucune API publique n'est utilisée/implémentée par ce projet; aucune API officielle publique n'a été vérifiée à la date du 2026-09-12.

Chapitre 3

Méthode 1 : URL iCal sécurisée (icalsecurise)

3.1 Principe général

Pronote expose un flux de calendrier en lecture seule conforme à la norme **iCalendar (RFC 5545)**. L'accès est contrôlé par un **jeton secret** intégré dans l'URL sous forme de paramètre de requête **icalsecurise**.

Statut :  Observé localement (fonctionnalité native de Pronote, version non spécifiée).

3.2 Obtention du jeton

1. Se connecter à Pronote (Espace Parents ou Élève) via n'importe quelle méthode.
2. Accéder à la vue « Emploi du temps ».
3. Utiliser la fonction « Export iCal » ou « Exporter ».
4. Pronote génère une URL contenant un jeton secret **icalsecurise**.
5. **Copier cette URL** : elle constitue une crédentiale unique sensible qui doit être protégée comme un mot de passe.

Source :  Observé localement dans les interfaces Pronote (version non spécifiée, hypothèse à valider).

Note : Les étapes dépendent de l'instance et du type de compte. L'exemple ci-dessus est non fonctionnel et ne contient aucun secret.

3.3 Format de l'URL

```
https://{etablissement}.index-education.net/pronote/ical/Edt_{prenom}.ics?icalsecurise={jeton}&version={version}&param={param}
```


Listing 3.1: Format d'URL iCal Pronote

- `icalsecurise` : **Jeton secret** (crédentiale).
- `version` : Version de Pronote (ex. 2024).
- `param` : Paramètres optionnels.

Source :  Analysé via `pronote_sync/sources/ical.py`.

3.4 Cycle de vie

- **Pérennité** : Le jeton reste valide jusqu'à :
 - Révocation manuelle par l'utilisateur dans Pronote.
 - Régénération par l'établissement (ex. à la rentrée scolaire).
-  **Hypothèse à valider** : La durée exacte dépend des politiques de l'établissement (non documentée officiellement).

Source :  Comportement variable selon les instances (à tester localement).

3.5 Sécurité

1. **Surface d'attaque** : Le jeton est encodé dans l'URL → risque d'exposition via :

- Logs serveur/proxy.
- En-tête Referer.
- Historique du navigateur.

2. **Recommandations** :

- **Ne jamais versionner** l'URL (ex. dans `.env` ou Git).
- Utiliser **HTTPS** (obligatoire).
- Masquer l'URL dans les logs (ex. via `redact_url()` dans `pronote-sync`).

Source :  Recommandation du projet (inspirée des bonnes pratiques générales de sécurité).

3.6 Intégration dans pronote-sync

- **Paramètre :** PRONOTE_ICAL_URL (ex. `.env.example` ligne 2).
- **Comportement :**
 - Prioritaire en mode PRONOTE_AGENDA_SOURCE=auto.
 - Si l'URL est invalide ou expire, repli automatique vers pronotepy (si PRONOTE_AGENDA_SOURCE=auto).
 - **Aucun repli** si PRONOTE_AGENDA_SOURCE=ical (échec explicite).

Source :  `pronote_sync/config/settings.py` + `pronote_sync/sources/ical.py`.

Chapitre 4

Méthode 2 : Connexion directe (identifiant / mot de passe + ENT)

4.1 Principe général

Connexion via le protocole propriétaire de Pronote (JSON sur HTTPS), avec **chiffrement AES-CBC utilisant une clé dérivée par MD5 (16 octets, soit AES-128)** comme implémenté dans pronotepy 2.15.7 et mécanisme de défi-réponse. Utilisé par l'interface web.

Statut :  Observé dans le code local (délégable à pronotepy via ParentClient ou Client).

4.2 Flux d'authentification

1. **Initialisation** : Requête GET vers `/pronote/{espace}.html` (ex. `parent.html`).
 - Récupère `h` (ID de session), `a` (ID espace), `sCrA/sCoA` (drapeaux chiffrement/compression).
2. **Échange de clés** : Requête POST vers `/pronote/appe fonction/{a}/{h}/{numeroOrdre}`.
3. **Identification** : Soumission de `identifiant`, `genreConnexion=0`, `genreEspace={a}`.
4. **Résolution du défi** :
 - Calcul de `mtp = MAJUSCULE(HEX(SHA256(alea + mot_de_passe)))`.
 - Dérivation de `key_challenge = MD5(nom_utilisateur + mtp)`.
 - Déchiffrement du challenge avec **AES-CBC utilisant une clé dérivée par MD5 (16 octets, soit AES-128)**.
5. **Authentification** : Soumission de la réponse au défi.

Source :  Observé dans le code local de pronotepy 2.15.7 (module `clients.py` et `pronoteAPI.py`, vérifié le 2026-09-12).

4.3 Intégration dans pronote - sync

— Paramètres :

- `PRONOTE_URL` (ex. `.env.example` ligne 3).
- `PRONOTE_USERNAME`, `PRONOTE_PASSWORD`.
- `PRONOTE_ENT` (slug dans `_ENT_NAMES`).
- `PRONOTE_ACCOUNT_TYPE` (ex. `parent`).

— Comportement :

- Utilise `pronotepy.ParentClient` pour les comptes parents.
- **Repli** : Si iCal échoue en mode auto, pronotepy est utilisé.
- **Pas de repli** si `PRONOTE_AGENDA_SOURCE=pronotepy` (échec explicite).

Source :  `pronote_sync/sources/pronote/client.py` (méthode `_connect_password`).

4.4 ENT supportés

Liste **fermée** des **30 ENT** résolubles (lignes 52-83 de `pronote_sync/sources/pronote/client.py`) :

```
_monbureaunumerique, ent_elyco, bordeaux, ent_creuse, occitanie_montpellier, ...
```

Listing 4.1: Liste des ENT supportés

Hypothèse à valider : Les ENT non listés nécessitent une contribution à pronotepy.

Source :  Code source local de pronote - sync (2026-09-12).

Chapitre 5

Méthode 3 : QR Code + token mobile (pour pronote-sync)

5.1 Principe général

Mécanisme d'appairage par QR code pour les appareils mobiles, **contournant l'authentification ENT/Edu-Connect**. Le QR code est généré **depuis l'interface web Pronote, espace parent → paramètres → QR code**, puis utilisé avec un **PIN à 4 chiffres** pour obtenir un token dont la durée dépend de l'instance/serveur.

Statut : ✓ Pris en charge par pronote-sync (mode PRONOTE_AUTH_MODE=qr_token).

5.2 Procédure QR pour pronote-sync

5.2.1 Étape 1 : Génération du QR code (interface web)

1. Se connecter à Pronote via un navigateur (espace **Parent**).
2. Aller dans **Paramètres → Accès mobile / Application mobile**.
3. Définir un **PIN temporaire à 4 chiffres**.
4. Pronote affiche un **QR code** contenant un JSON avec les clés :
 - login, jeton, et url (ex. `https://[host]/pronote/mobile.parent.html`).

Le fichier JSON réel contient des identifiants chiffrés et ne doit jamais être copié, partagé, ou committé.

5. Exporter le QR code :

- Sauvegardez le fichier JSON localement ou scannez-le avec un appareil.

- **Ne jamais partager** le JSON ou le PIN.
- **Le fichier JSON du QR code contient des identifiants chiffrés : ne jamais le coller dans la documentation ni le committer.**

Source :  Observé localement dans l'interface web Pronote (version non spécifiée, hypothèse à valider).

5.2.2 Étape 2 : Configuration de pronote-sync

1. Enregistrer le QR code :

- Sauvegarder le JSON dans un fichier (ex. `/path/to/qr_code.json`).
- **Permissions :** `chmod 600 /path/to/qr_code.json`.

2. Configurer `.env` :

```
PRONOTE_AUTH_MODE=qr_token
PRONOTE_QR_CODE_FILE=/path/to/qr_code.json
PRONOTE_QR_PIN= # renseigner localement la valeur secrète du PIN (jamais committée)
```

Listing 5.1: Configuration QR Code dans `.env`

Note : `.env.example` ne doit **jamais** contenir de PIN concret.

Source :  Observé dans `.env.example` (lignes 21-25) et `pronote_sync/sources/pronote/client.py`.

5.2.3 Étape 3 : Premier login (enrôlement)

1. `pronote-sync` lit `PRONOTE_QR_CODE_FILE` et `PRONOTE_QR_PIN`.
2. Appel à `pronotepy.ParentClient.qrcode_login(qr_code, pin, uuid)` :
 - `qr_code` : JSON du fichier QR.
 - `pin` : PIN à 4 chiffres.
 - `uuid` : UUID permanent généré par `pronote-sync` (ex. `pronote-sync-{uuid4()}`).
3. **Déchiffrement :**
 - Algorithme : **AES-CBC utilisant une clé dérivée par MD5 (16 octets, soit AES-128).**
 - Clé : MD5 (PIN) (dérivée du PIN secret).

- IV : 16 octets nuls.
 - Résultat : login et jeton en clair.
4. **Échange de token** : Pronote retourne un `jetonConnexionAppliMobile` (token dont la durée dépend de l'instance/serveur).
 5. **Persistence** : `pronote-sync` sauvegarde les credentials via `export_credentials()` dans `.pronote_auth_state` (mode 0600).

Source :  Observé dans le code local de `pronotepy 2.15.7` (`clients.py` lignes 181–189) + `pronote_sync/sources/pronote/client.py` (méthode `_enroll_qr_code`).

5.2.4 Étape 4 : Connexions ultérieures (auto-login)

1. `pronote-sync` charge `.pronote_auth_state.json`.
2. Appel à `pronotepy.ParentClient.token_login(**credentials)` :
 - `pronote_url`, `username`, `password (token)`, `uuid`.
3. **Rotation du token** : Le token est **remplacé uniquement si le serveur renvoie `jetonConnexionAppliMobile`** (observé dans `pronotepy 2.15.7`, `clients.py`:382–387).
4. **Persistence** : Les credentials sont sauvegardés dans `.pronote_auth_state.json` après chaque opération réussie.

Source :  Observé dans le code local de `pronotepy 2.15.7` (`clients.py` lignes 245–280) + `pronote_sync/sources/pronote/client.py` (méthode `_connect_qr_token`).

5.3 Cycle de vie des tokens

- **QR code** : Valide **10 minutes** après génération ( **Hypothèse à valider** : cette durée n'est attestée que par un message d'exception dans `pronotepy` et n'est pas une garantie officielle/indépendante de l'instance).
- **jetonConnexionAppliMobile** :
 - Durée **dépendante de l'instance/serveur** (observation du 2026-09-12, hypothèse : peut persister jusqu'à la fin de l'année scolaire, non garanti).
 - **Remplacé uniquement si le serveur renvoie `jetonConnexionAppliMobile`** (observé dans `pronotepy 2.15.7`, `clients.py`:382–387).

- **Révocable** manuellement dans Pronote (Paramètres→Accès mobile).

Source : 📖 Comportement variable selon les instances (à tester localement).

5.4 Sécurité

1. Fichiers sensibles :

- `.pronote_auth_state.json` : **Ne jamais versionner** (couvert par `.gitignore`).
- `PRONOTE_QR_CODE_FILE` : **Ne jamais committer** (ex. dans Git).

2. Secrets :

- Le PIN et le contenu du QR code sont **masqués** dans les logs (via `redact_secrets()`).
- Les exceptions sont **expurgées** (via `redact_exception()`).

3. Recommandations :

- Utiliser un **PIN robuste** (éviter les codes simples comme « 0000 » ou des séquences évidentes).
- **Révoquer** le token en cas de compromission (via Pronote web).

Source : 📖 `pronote_sync/utils/redaction.py` + `pronote_sync/sources/pronote/client.py` (méthode `collect_auth_secrets`).

5.5 Erreurs et repli

- **PronoteAuthRotationError** : Levée si :

- Le token persisté est **invalide/expiré**.
- Le fichier QR ou le PIN est **manquant/invalide**.

- **Action requise :**

1. Supprimer `.pronote_auth_state.json`.
2. Générer un **nouveau QR code depuis l'interface web Pronote, espace parent**.
3. Relancer `pronote-sync`.

Source :  Observé dans `pronote_sync/errors.py` + `pronote_sync/sources/pronote/client.py` (lignes 346-364).

5.6 Incompatibilités

- **Mode dry-run :** **Incompatible** avec `PRONOTE_AUTH_MODE=qr_token` (risque de désynchronisation du token local).
- `pronote-sync --dry-run` **refuse** le mode `qr_token` avant toute connexion.

Source :  `docs/exploitation.md` (ligne 65-66).

Chapitre 6

Méthode 4 : SSO ENT (CAS/SAML)

Statut : X Non implémenté dans pronote - sync.

Seuls les 30 ENT de la liste fermée _ENT_NAMES (pronotepy 2.15.7) sont supportés via ent dans ParentClient. **Pas de SSO générique CAS/SAML.**

Chapitre 7

Méthode 5 : EduConnect (HubEduConnect)

Statut : ✗ Non implémenté dans pronote - sync.

Nécessite une intégration CAS/SAML générique, **non supportée par ce projet.**

Chapitre 8

Méthode 6 : CAS direct

Statut : ✗ Non implémenté dans pronote - sync.

Non supporté.

Chapitre 9

Méthode 7 : API publique

Statut : ✗ Inexistante.

Aucune API publique n'est utilisée/implémentée par ce projet ; aucune API officielle publique n'a été vérifiée à la date du 2026-09-12.

Annexe A

Bibliothèques tierces

Table A.1 : Bibliothèques tierces et leur statut dans pronote - sync

Bibliothèque	Langage	Dépôt	Méthodes supportées	Statut dans pronote-sync
pronotepy	Python	https://github.com/bain3/pronotepy	Connexion directe, QR code/token, 30 ENT (liste fermée)	✓ Dépendance principale (version 2.15.7 vérifiée).
pronote-api	TypeS-crypt	https://github.com/Litarvan/pronote-api	Connexion directe, CAS, ENT	✗ Non utilisée.
pronote-qr-code-api	JavaS-crypt	https://github.com/Androz2091/pronote-qr-code-api	Déchiffrement QR code	✗ Non utilisée (intégration native via pronotepy).

Source :  Observé dans pyproject.toml (dépendances) + code source local (2026-09-12).

Annexe B

Tableau comparatif synthétique

Table B.1 : Comparatif des méthodes d'authentification

Méthode	Périmètre	Identifiants	Expiration	Complexité	Statut dans pronote-sync
URL iCal sécurisée	EDT + devoirs (si inclus dans le flux)	Jeton URL	Longue durée (dépend de l'instance)	Très faible	✓ Pris en charge
Connexion directe (mot de passe + ENT)	Agenda/EDT, devoirs, messages	Identifiant + mot de passe + ENT	Session dépendante de l'instance	Élevée	✓ Pris en charge
QR Code + token mobile	Agenda/EDT, devoirs, messages	PIN 4 chiffres → token + UUID	QR 10 min (138 hypothèse); token dépendant de l'instance	Modérée	✓ Pris en charge
SSO ENT (CAS/SAML)	—	Identifiants ENT	Session ENT	Très élevée	✗ Non implémenté (seuls les 30 ENT de la liste fermée _ENT_NAMES sont supportés via pronotepy)
EduConnect	—	Identifiants nationaux	Session EduConnect	Très élevée	✗ Non implémenté
CAS direct	—	Identifiants CAS	Ticket single-use	Modérée	✗ Non implémenté
API publique	N/A	N/A	N/A	N/A	✗ Inexistante

Légende

- **Périmètre** : Les méthodes **iCal** sont en lecture seule (EDT + devoirs si inclus dans le flux). Les méthodes **Connexion directe** et **QR Code + token mobile** permettent les opérations pronotepy effectivement implémentées par ce projet (agenda, devoirs, messages ; informations ignorées en mode qr_token).
- **QR 10 min** : 🚫 **Hypothèse non vérifiée** (observé dans pronotepy via un message d'exception, dépend de l'instance).
- **Session dépendante de l'instance** : 🚫 **Hypothèse non vérifiée** (nécessite un essai réel).

Annexe C

Écarts et notes de cohérence

C.1 Alignement avec `.env.example`

Table C.1 : Alignement avec `.env.example`

Paramètre	Document	Code	Statut
PRONOTE_ICAL_URL	✓ Lignes 2, 42-50	✓ <code>sources/ical.py</code>	Cohérent
PRONOTE_URL	✓ Ligne 3	✓ <code>client.py</code> (ligne 294)	Cohérent
PRONOTE_ENT	✓ Ligne 7	✓ <code>client.py</code> (ligne 297, <code>_ENT_NAMES</code>)	Cohérent
PRONOTE_AUTH_MODE=qr_token	✓ Ligne 23	✓ <code>client.py</code> (ligne 334)	Cohérent
PRONOTE_QR_CODE_FILE	✓ Ligne 24	✓ <code>client.py</code> (ligne 387)	Cohérent
PRONOTE_QR_PIN	✓ Ligne 25	✓ <code>client.py</code> (ligne 388)	Cohérent
« Le QR code se génère sur le site web »	✓ Ligne 21	✓ Section « Procédure QR pour pronote-sync »	Cohérent

C.2 Écarts avec `docs/exploitation.md`

Table C.2 : Écarts avec `docs/exploitation.md`

Élément		Ce document	Action
	exploitation.md		
« Le QR code expire ~10 minutes »	Ligne 22	⚠ Hypothèse non vérifiée (section « Cycle de vie des tokens »)	Aucune (hors périmètre).
« Mode <code>qr_token</code> incompatible avec <code>dry-run</code> »	Lignes 65-66	✓ Section « Incompatibilités »	Cohérent
« Fichier <code>.pronote_auth_state.json</code> (mode <code>o600</code>) »	Lignes 70-75	✓ Section « Sécurité »	Cohérent

Glossaire

Terme	Définition
ENT	Espace Numérique de Travail (ex. Mon Bureau Numérique, Paris Classe Numérique).
Jeton icalsecurise	Token secret intégré dans l'URL iCal, équivalent à un mot de passe.
jetonConnexionAppliMobile	Token obtenu après appairage QR code, dont la durée dépend de la durée de la connexion/du serveur et n'est pas garantie.
UUID	Identifiant unique permanent pour l'application (ex. pronote-sync-{uuid4()}).
PIN	Code à 4 chiffres défini lors de la génération du QR code.

Historique des révisions

Date	Auteur	Modifications
2026-09-12	Agent tech-writer	Refonte complète : qualification des affirmations, ajout des sources, tableau de compatibilité, procédure QR alignée sur pronotepy 2.15.7.
2026-09-12	Agent tech-writer	Corrections suite à revue indépendante (ticket #10) : précision cryptographie (AES-128), qualification des sources, alignement nombre d'ENT (30), clarification SSO/CAS, exemples non copiables, note d'essai réel.
2026-09-12	Agent tech-writer	Corrections suite à revue ticket #10 : suppression de tous les placeholders de secrets remplacés par de la prose descriptive ; clarification du périmètre réel des méthodes d'authentification (accès limité aux opérations implémentées) ; qualification de la durée du token <code>jetonConnexionAppliMobile</code> comme dépendante de l'instance/serveur et non garantie.