API de synchronisation et SSO OAuth2

Documentation développeurs pour les réseaux, sièges et portails partenaires

Ce document décrit le contrat d'échange serveur-à-serveur pour les agences et utilisateurs, ainsi que le parcours de connexion SSO des boutiques et du back-office.

Publication	11 juin 2026
Périmètre	POST /api/network/sync • OAuth2 Authorization Code
Statut	Contrat d'intégration actuel

Document destiné aux équipes techniques tierces. Les domaines, clés et identifiants utilisés dans les exemples sont fictifs et doivent être remplacés par les valeurs communiquées pour votre réseau.

1. Démarrage rapide

Deux mécanismes complémentaires sont disponibles. La synchronisation crée et maintient les comptes ; le SSO permet ensuite aux utilisateurs autorisés de se connecter sans mot de passe local.

Mécanisme	Point d'entrée	Authentification	Usage
`Synchronisation`	POST /api/network/sync	Clé API Bearer	Agences, utilisateurs, rôles, accès boutiques et sujets SSO
`SSO boutique`	GET /connexion/sso/start	OAuth2 Authorization Code	Connexion depuis une boutique du réseau
`SSO back-office`	GET /auth/sso/start	OAuth2 Authorization Code	Connexion des profils disposant d'un accès d'administration

1.1 Ordre d'intégration recommandé

1. Obtenir l'URL de plateforme, la clé API du réseau, le code fournisseur SSO et les codes boutiques.
2. Configurer le fournisseur OAuth2 et enregistrer exactement les URL de callback communiquées.
3. Tester une synchronisation partial avec une agence et un utilisateur de recette.
4. Vérifier le lien OAuth créé à partir de sso_subject, puis tester la connexion depuis la boutique.
5. Automatiser les exports partial ; n'activer full qu'après validation d'un export exhaustif.

> **Principe d'identité.** L'identifiant pivot de la synchronisation est le couple réseau + external_id. L'adresse e-mail ne sert jamais à rattacher automatiquement un nouvel external_id à un compte existant.

1.2 Valeurs à obtenir avant la recette

- URL de base de la plateforme, par exemple https://plateforme.exemple.fr.
- Clé API réseau au format net_... ; elle n'est affichée qu'une fois à sa création.
- Codes des boutiques actives autorisées pour le réseau.
- Code du fournisseur SSO, client_id, secret, endpoints authorize/token/userinfo et scopes.
- URL exactes de callback boutique et, si nécessaire, back-office.

2. API de synchronisation réseau

2.1 Requête HTTP

```
POST https://<domaine-plateforme>/api/network/sync
Authorization: Bearer net_<secret>
Content-Type: application/json
```

La clé est liée à un seul réseau. Toutes les agences, boutiques, utilisateurs, fournisseurs SSO et rôles référencés sont donc contrôlés dans ce périmètre.

> **Secret sensible.** Conservez la clé dans un coffre de secrets, ne l'ajoutez jamais dans une URL et ne la journalisez pas. Une clé révoquée doit être remplacée immédiatement dans l'intégration.

2.2 Contraintes globales

Contrôle	Valeur par défaut	Comportement
`Taille du corps`	1 Mio	413 payload_too_large au-delà de la limite
`Nombre d'objets`	5 000	Somme de locations + users ; 413 au-delà
`Quota`	60 requêtes / heure / clé	429 avec details.retry_after_seconds
`Mode full`	30 % maximum	Blocage si plus de 30 % des actifs seraient désactivés
`Structure JSON`	Stricte	Tout champ inconnu provoque une erreur 422

Les limites peuvent être adaptées par l'exploitant. Une intégration ne doit pas supposer qu'elles seront toujours égales aux valeurs par défaut.

2.3 Objet racine

Champ	Obligatoire	Type / valeurs	Description
`sync_mode`	Oui	partial | full	Mode de traitement de la requête.
`locations`	Full : oui	Liste d'objets	Agences ou implantations du réseau.
`users`	Full : oui	Liste d'objets	Utilisateurs et droits à provisionner.

En mode partial, une liste absente est interprétée comme une liste vide et n'entraîne aucune désactivation des objets non transmis. En mode full, locations et users doivent toutes deux être présentes.

2.4 Exemple minimal partial

```
{
  "sync_mode": "partial",
  "locations": [
    {
      "external_id": "AG-001",
      "name": "Agence Lyon",
      "city": "Lyon",
      "status": "active",
      "shop_codes": ["LYON-PRO"]
    }
  ],
  "users": [
    {
      "external_id": "USR-10042",
      "firstname": "Camille",
      "lastname": "Martin",
      "email": "camille.martin@partenaire.fr",
      "role": "ROLE_LOCATION_MANAGER",
      "status": "active",
      "location_external_id": "AG-001",
      "sso_subject": "00u8abc123xyz",
      "sso_provider_code": "partenaire-pro"
    }
  ]
}
```

3. Référence des agences

Champ	Requis	Contraintes	Effet
`external_id`	Oui	Texte, 190 car. max	Identifiant stable et unique dans le réseau.
`name`	Oui	Texte, 190 car. max	Nom affiché de l'agence.
`address`	Non	Texte, 255 car. max	Adresse postale.
`postal_code`	Non	Texte, 30 car. max	Code postal.
`city`	Non	Texte, 120 car. max	Ville.
`phone`	Non	Texte, 40 car. max	Téléphone.
`email`	Non	E-mail valide, 190 car. max	Adresse de contact de l'agence.
`status`	Non	active | inactive	Valeur par défaut : active.
`shop_codes`	Non	Liste de textes	Remplace les boutiques liées lorsqu'il est présent.

3.1 Sémantique de shop_codes

- Champ absent : les liaisons agence-boutiques existantes sont conservées.
- Liste non vide : elle remplace la liste actuelle de boutiques de l'agence.
- Liste vide [] : toutes les liaisons agence-boutiques sont supprimées.
- Chaque code doit désigner une boutique active appartenant au réseau de la clé API.

> **Attention aux listes vides.** shop_codes: [] est une instruction explicite de retrait des accès hérités de cette agence. Omettez le champ lorsque vous ne souhaitez pas modifier les liaisons.

3.2 Mise à jour et statut

Un même external_id met à jour l'agence existante. status: inactive désactive l'agence sans supprimer physiquement son historique. Le passage ultérieur à active la réactive.

```
{
  "sync_mode": "partial",
  "locations": [
    {
      "external_id": "AG-001",
      "name": "Agence Lyon Centre",
      "address": "18 rue Exemple",
      "postal_code": "69002",
      "city": "Lyon",
      "phone": "+33 4 00 00 00 00",
      "email": "lyon@partenaire.fr",
      "status": "active",
      "shop_codes": ["LYON-PRO", "CATALOGUE-NATIONAL"]
    }
  ]
}
```

4. Référence des utilisateurs

Champ	Requis	Contraintes	Effet
`external_id`	Oui	Texte, 190 car. max	Identifiant stable et unique dans le réseau.
`firstname`	Oui	Texte, 120 car. max	Prénom du compte local.
`lastname`	Oui	Texte, 120 car. max	Nom du compte local.
`email`	Oui	E-mail valide, 190 car. max	Adresse du compte ; normalisée en minuscules.
`role`	Non	Texte, 190 car. max	Rôle externe ; défaut ROLE_LOCATION_USER.
`status`	Non	active | inactive	Défaut active ; inactive retire les habilitations.
`location_external_id`	Non	Texte, 190 car. max	Agence du même réseau.
`shop_codes`	Non	Liste de textes	Accès boutiques direct ; prioritaire sur l'héritage agence.
`sso_subject`	Non	Texte, 190 car. max	Identifiant stable renvoyé par le fournisseur OAuth.
`sso_provider_code`	Non	Texte, 100 car. max	Code du fournisseur SSO actif du réseau.

4.1 Identité et collisions

- La clé métier est réseau + external_id.
- Un nouvel external_id dont l'e-mail appartient déjà à un autre compte est bloqué afin d'éviter une prise de contrôle.
- Les external_id dupliqués dans une même requête sont rejetés.
- Les changements de prénom, nom, e-mail, rôle, statut et agence mettent à jour le compte provisionné.
- Un mot de passe aléatoire est généré à la création ; la connexion attendue est le SSO.

4.2 Accès boutiques

Situation	Boutiques attribuées
`shop_codes présent`	La liste fournie est appliquée directement à l'utilisateur.
`shop_codes absent + agence renseignée`	L'utilisateur hérite des boutiques actuellement liées à l'agence.
`shop_codes absent + aucune agence`	Aucune boutique n'est attribuée pour un rôle de portée boutique.
`status inactive`	Les habilitations provisionnées du réseau sont désactivées.

> **Réactivation.** Un utilisateur revenu dans un export avec status: active est réactivé. Les habilitations sont recalculées à partir du rôle, de shop_codes et de l'agence transmis.

5. Rôles et pré-provisionnement SSO

5.1 Mappage des rôles par défaut

Rôle externe	Portée	Rôle plateforme
`ROLE_NETWORK_ADMIN`	Réseau	network_manager
`ROLE_LOCATION_MANAGER`	Boutique	agency_manager
`ROLE_LOCATION_USER`	Boutique	agency_agent
`ROLE_MARKETING`	Réseau	independent_agent
`ROLE_READONLY`	Boutique	agency_agent
`Toute autre valeur`	Boutique	agency_agent

L'exploitant peut surcharger ces correspondances pour un réseau. Le partenaire doit donc valider les rôles attendus pendant la recette, même s'il utilise les valeurs standard ci-dessus.

5.2 Création du lien SSO pendant la synchronisation

Lorsque sso_subject est fourni et non vide, la plateforme tente de créer le lien entre l'utilisateur local et le compte externe du fournisseur OAuth.

- Si sso_provider_code est fourni, il doit désigner un fournisseur actif du même réseau.
- S'il est omis, le réseau doit posséder exactement un fournisseur SSO actif.
- S'il existe zéro ou plusieurs fournisseurs actifs, sso_provider_code devient obligatoire.
- L'omission ultérieure des champs SSO conserve les valeurs déjà enregistrées.
- Un même fournisseur + sujet ne peut jamais être lié à deux utilisateurs.
- Un conflit de lien annule toute la synchronisation en cours.

> **Sujet OAuth.** Utilisez un identifiant immuable et opaque, typiquement la claim sub. Ne réutilisez pas l'e-mail comme sujet : il peut changer et n'offre pas les mêmes garanties d'unicité.

5.3 Exemple utilisateur complet

```
{
  "external_id": "USR-10042",
  "sso_subject": "00u8abc123xyz",
  "sso_provider_code": "partenaire-pro",
  "firstname": "Camille",
  "lastname": "Martin",
  "email": "camille.martin@partenaire.fr",
  "role": "ROLE_LOCATION_MANAGER",
  "status": "active",
  "location_external_id": "AG-001",
  "shop_codes": ["LYON-PRO"]
}
```

6. Modes partial et full

6.1 Mode partial

partial réalise des créations et mises à jour ciblées. Les objets absents de la requête ne sont pas désactivés. C'est le mode conseillé pour les événements unitaires, les reprises et la recette.

- locations et users peuvent être omis indépendamment.
- status: inactive désactive explicitement l'objet transmis.
- Une requête répétée avec les mêmes external_id converge vers le même état fonctionnel.
- Aucune clé d'idempotence HTTP n'est attendue ; chaque appel reçoit un nouveau request_id.

6.2 Mode full

full représente un état exhaustif du réseau. Après les mises à jour, toute agence ou tout utilisateur actif absent de la liste correspondante est désactivé. Aucun enregistrement n'est supprimé physiquement.

> **Opération à fort impact.** Un export full incomplet peut retirer des accès. Construisez le fichier dans une zone temporaire, contrôlez ses volumes, puis envoyez-le seulement lorsque locations et users sont complets.

6.3 Garde-fou de désactivation

Avant toute modification, la plateforme compare le nombre d'objets actifs qui deviendraient inactifs avec le nombre actuel d'actifs, séparément pour users et locations.

```
ratio = objets_actifs_qui_deviendraient_inactifs / objets_actifs_actuels

si ratio > 0,30 : rejet complet de la synchronisation
si ratio <= 0,30 : traitement autorisé
```

Les objets transmis explicitement avec status: inactive comptent parmi les objets qui deviendraient inactifs. Au moindre dépassement, aucune agence ni aucun utilisateur n'est modifié.

6.4 Transactions et cohérence

- Les créations, mises à jour, liens SSO et désactivations d'une requête sont transactionnels.
- Une erreur sur un seul objet annule l'ensemble des modifications de la requête.
- Le journal de synchronisation conserve le statut, les volumes et le request_id à des fins de support.
- Recommandation d'exploitation : sérialiser les synchronisations d'un même réseau.

7. Réponses et erreurs

7.1 Réponse de succès

```
{
  "ok": true,
  "request_id": "7f5d186c3f034db5b6eb94eb1ec2b468",
  "sync_log_id": 31,
  "network_id": 12,
  "sync_mode": "partial",
  "stats": {
    "received_locations": 1,
    "received_users": 1,
    "created_locations": 1,
    "updated_locations": 0,
    "deactivated_locations": 0,
    "created_users": 1,
    "updated_users": 0,
    "deactivated_users": 0,
    "errors": []
  }
}
```

7.2 Réponse d'erreur

```
{
  "ok": false,
  "request_id": "9946b612e2f2442db27267c240ae5ad2",
  "error": {
    "code": "validation_error",
    "message": "users[0].email est invalide.",
    "details": {}
  }
}
```

details est omis lorsqu'il est vide. Conservez systématiquement request_id dans les journaux de votre connecteur et transmettez-le au support lors d'une investigation.

7.3 Codes HTTP

HTTP	Code	Action intégrateur
400	invalid_json	Corriger la syntaxe ou le corps JSON vide ; ne pas relancer tel quel.
401	unauthorized	Vérifier ou renouveler la clé Bearer.
413	payload_too_large	Réduire la taille ou le nombre d'objets.
415	unsupported_media_type	Envoyer Content-Type: application/json.
422	validation_error	Corriger les données ou la cohérence métier.
429	rate_limited	Attendre details.retry_after_seconds avant une nouvelle tentative.
500	internal_error	Réessayer avec backoff ; escalader avec request_id si persistant.

7.4 Politique de reprise recommandée

- Ne jamais relancer automatiquement une erreur 400, 401, 413, 415 ou 422 sans correction.
- Pour 429, respecter le délai retry_after_seconds et ajouter une petite gigue aléatoire.
- Pour 500 ou une erreur réseau, utiliser un backoff exponentiel borné.
- Après une réponse ambiguë, relire votre source puis renvoyer le même état en partial ; l'upsert se base sur external_id.

8. Exemples d'appel

8.1 cURL

```
curl --request POST "https://plateforme.exemple.fr/api/network/sync" \
  --header "Authorization: Bearer ${NETWORK_SYNC_API_KEY}" \
  --header "Content-Type: application/json" \
  --data-binary @sync.json
```

8.2 PHP 8.2+

```
<?php

$endpoint = 'https://plateforme.exemple.fr/api/network/sync';
$apiKey = getenv('NETWORK_SYNC_API_KEY');
$payload = [
    'sync_mode' => 'partial',
    'users' => [[
        'external_id' => 'USR-10042',
        'firstname' => 'Camille',
        'lastname' => 'Martin',
        'email' => 'camille.martin@partenaire.fr',
        'status' => 'active',
    ]],
];

$curl = curl_init($endpoint);
curl_setopt_array($curl, [
    CURLOPT_POST => true,
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_TIMEOUT => 30,
    CURLOPT_HTTPHEADER => [
        'Authorization: Bearer ' . $apiKey,
        'Content-Type: application/json',
    ],
    CURLOPT_POSTFIELDS => json_encode($payload, JSON_THROW_ON_ERROR),
]);

$body = curl_exec($curl);
$status = curl_getinfo($curl, CURLINFO_RESPONSE_CODE);
if ($body === false) {
    throw new RuntimeException(curl_error($curl));
}
$result = json_decode($body, true, 64, JSON_THROW_ON_ERROR);
if ($status >= 400 || ($result['ok'] ?? false) !== true) {
    throw new RuntimeException(
        ($result['error']['message'] ?? 'Erreur API')
        . ' request_id=' . ($result['request_id'] ?? 'inconnu')
    );
}
```

8.3 Node.js 20+

```
const endpoint = 'https://plateforme.exemple.fr/api/network/sync';
const payload = {
  sync_mode: 'partial',
  users: [{
    external_id: 'USR-10042',
    firstname: 'Camille',
    lastname: 'Martin',
    email: 'camille.martin@partenaire.fr',
    status: 'active'
  }]
};

const response = await fetch(endpoint, {
  method: 'POST',
  headers: {
    Authorization: `Bearer ${process.env.NETWORK_SYNC_API_KEY}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(payload),
  signal: AbortSignal.timeout(30_000)
});
const result = await response.json();
if (!response.ok || result.ok !== true) {
  throw new Error(
    `${result.error?.message ?? 'Erreur API'} request_id=${result.request_id}`
  );
}
```

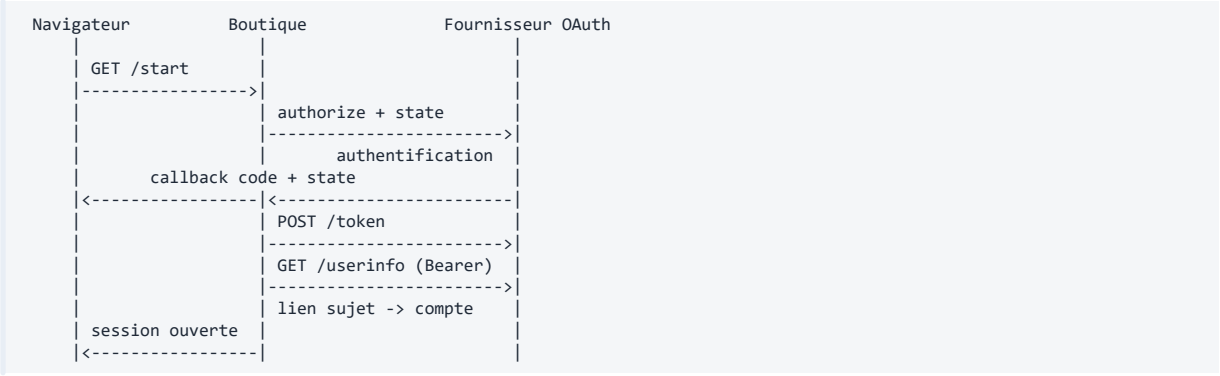
9. SSO OAuth2 : architecture

La plateforme agit comme client OAuth2. Le partenaire héberge le fournisseur d'identité et expose les endpoints d'autorisation, d'échange de code et de profil utilisateur.

9.1 Capacités attendues du fournisseur

- Flux OAuth2 Authorization Code.
- Endpoint authorize acceptant response_type, client_id, redirect_uri, scope et state.
- Endpoint token acceptant un formulaire application/x-www-form-urlencoded.
- Réponse token JSON contenant access_token.
- Endpoint userinfo JSON appelé avec Authorization: Bearer <access_token>.
- Claim utilisateur stable et non vide, généralement sub.

9.2 Séquence de connexion boutique



9.3 URL de démarrage

```
https://<domaine-boutique>/connexion/sso/start
  ?provider=<code-fournisseur>
  &redirect=/account
```

redirect est facultatif. Seuls les chemins internes sont acceptés ; la valeur par défaut est /account. Les chemins d'administration et de callback SSO sont refusés.

10. Callbacks et paramètres OAuth

10.1 URL à enregistrer chez le fournisseur

Canal	Callback exact
`Boutique`	https://<domaine-boutique>/connexion/sso/callback?provider=<code>
`Back-office`	https://<domaine-admin>/auth/sso/callback?provider=<code>

Correspondance exacte. La redirect_uri envoyée au token est identique à celle de l'autorisation. Pour plusieurs domaines de boutique, chaque URL exacte doit être autorisée selon les capacités du fournisseur.

10.2 Requête d'autorisation construite par la plateforme

```
GET https://idp.partenaire.fr/oauth2/authorize
  ?response_type=code
  &client_id=<client_id>
  &redirect_uri=<callback_exact>
  &scope=openid%20profile%20email
  &state=<valeur_aleatoire>
```

10.3 Échange du code

```
POST https://idp.partenaire.fr/oauth2/token
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&client_id=<client_id>
&client_secret=<client_secret>
&redirect_uri=<callback_exact>
&code=<authorization_code>
```

La réponse doit être un objet JSON contenant access_token. Les champs supplémentaires sont tolérés mais ne sont pas requis par ce contrat.

10.4 Appel userinfo

```
GET https://idp.partenaire.fr/oauth2/userinfo
Authorization: Bearer <access_token>
Accept: application/json

{
  "sub": "00u8abc123xyz",
  "email": "camille.martin@partenaire.fr",
  "email_verified": true,
  "given_name": "Camille",
  "family_name": "Martin"
}
```

11. Configuration du fournisseur SSO

Paramètre	Requis	Défaut	Description
`code`	Oui	-	Code stable utilisé dans les URL et la synchronisation.
`name`	Oui	-	Libellé affiché.
`client_id`	Oui	-	Identifiant OAuth de la plateforme.
`client_secret`	Oui	-	Secret OAuth conservé côté serveur.
`authorize_url`	Oui	-	Endpoint d'autorisation.
`token_url`	Oui	-	Endpoint d'échange du code.
`userinfo_url`	Oui	-	Endpoint de profil JSON.
`scopes`	Non	openid profile email	Scopes séparés par des espaces.
`user_id_field`	Non	sub	Chemin de la claim identifiant le compte externe.
`email_field`	Non	email	Chemin de la claim e-mail.
`email_verified_field`	Non	email_verified	Claim utilisée pour sécuriser l'auto-liaison.
`first_name_field`	Non	given_name	Claim prénom, informative.
`last_name_field`	Non	family_name	Claim nom, informative.
`allowed_email_domains`	Non	-	Domaines exacts, séparés par espace, virgule ou point-virgule.
`auto_link_by_email`	Non	désactivé	Autorise le premier lien par e-mail sous contraintes.
`is_enabled`	Non	activé	Rend le fournisseur utilisable dans son réseau.

Les champs de claims acceptent un chemin avec points. Par exemple profile.identifiers.subject lit la valeur JSON imbriquée correspondante.

11.1 Règles d'auto-liaison par e-mail

- L'auto-liaison doit être explicitement activée.
- Au moins email_verified_field ou allowed_email_domains doit être configuré.
- Si les deux sont configurés, les deux contrôles doivent réussir.
- La claim vérifiée accepte true, 1, "true", "yes" ou "oui".
- Le domaine doit correspondre exactement après normalisation en minuscules.
- La recherche de l'utilisateur actif reste limitée au réseau du fournisseur.

> **Option la plus robuste.** Pré-provisionnez sso_subject par l'API de synchronisation. L'auto-liaison par e-mail est utile pour une transition, mais dépend de la qualité et de la vérification des claims du fournisseur.

12. Résolution du compte et contrôle d'accès

12.1 Ordre de résolution

1. La plateforme recherche un lien existant par fournisseur + sujet externe.
2. À défaut, si l'auto-liaison est activée, elle valide l'e-mail vérifié et/ou son domaine.
3. Elle recherche un utilisateur local actif du même réseau par e-mail.
4. Elle crée le lien OAuth sans écraser un lien conflictuel.
5. Elle ouvre la session puis vérifie l'accès à la boutique demandée.

Le SSO ne crée jamais un compte à lui seul. Le compte doit avoir été créé auparavant, en pratique par l'API de synchronisation ou par l'administration de la plateforme.

12.2 Accès à une boutique

Après authentification, l'accès est accordé si l'utilisateur possède au moins l'une des habilitations actives suivantes :

- une adhésion directe à la boutique ;
- un rôle opérateur plateforme super_admin, admin ou master_admin ;
- une adhésion network_manager au réseau de la boutique.

Sinon, la session est immédiatement fermée et l'utilisateur est renvoyé vers la page de connexion avec un message générique.

12.3 Protection du parcours

- state contient 32 octets aléatoires encodés en hexadécimal.
- Le state est lié en session au fournisseur, au réseau et à la boutique.
- Le contexte de handshake est effacé après le callback, qu'il réussisse ou échoue.
- La redirect_uri est reconstruite depuis le domaine courant de la boutique.
- La destination finale est limitée aux chemins internes sûrs.

12.4 Événements d'audit

Événement	Signification
`front_sso_start`	Début d'un parcours OAuth depuis une boutique.
`front_sso_callback_success`	Connexion et contrôle d'accès réussis.
`front_sso_callback_failed`	Échec de state, code, liaison, politique e-mail ou accès.
`front_sso_user_link_created`	Création d'un nouveau lien fournisseur-sujet-utilisateur.

13. Sécurité et exploitation

13.1 Secrets et journaux

- Utiliser HTTPS en production pour tous les endpoints.
- Stocker clé API et client_secret dans un gestionnaire de secrets.
- Ne jamais journaliser le header Authorization, le code OAuth, access_token ou client_secret.
- Journaliser request_id, statut HTTP, durée, volumes reçus et identifiants external_id concernés.
- Révoquer une clé API compromise depuis l'administration du réseau.

13.2 Validation des données

- Conserver external_id et sso_subject stables dans le temps.
- Valider les e-mails, rôles, statuts et codes boutiques avant l'envoi.
- Ne pas ajouter de champs libres : le contrat rejette les propriétés inconnues.
- Pour full, comparer les volumes avec le dernier export réussi avant l'appel.
- Traiter une réponse HTTP 200 comme un succès uniquement si ok vaut true.

13.3 Paramètres d'exploitation côté plateforme

Variable	Défaut	Rôle
`NETWORK_SYNC_MAX_BODY_BYTES`	1048576	Taille maximale du JSON.
`NETWORK_SYNC_MAX_OBJECTS`	5000	Nombre total locations + users.
`NETWORK_SYNC_RATE_LIMIT`	60	Nombre d'appels dans la fenêtre.
`NETWORK_SYNC_RATE_LIMIT_WINDOW_SECONDS`	3600	Durée de la fenêtre de quota.
`NETWORK_SYNC_MAX_FULL_DEACTIVATION_RATIO`	0.30	Seuil de blocage du mode full.

Ces paramètres sont administrés par l'exploitant de la plateforme ; ils sont fournis ici pour expliquer les comportements observables, pas comme prérequis de configuration côté partenaire.

14. Plan de recette

14.1 Synchronisation

- Créer une agence active et vérifier ses boutiques.
- Créer un utilisateur actif lié à l'agence et vérifier son rôle.
- Renvoyer les mêmes objets et vérifier l'absence de doublon.
- Modifier nom, e-mail, rôle, agence et shop_codes.
- Désactiver puis réactiver un utilisateur.
- Tester un champ inconnu, un e-mail invalide et un external_id dupliqué.
- Tester un conflit d'e-mail entre deux external_id.
- Tester un full volontairement incomplet et vérifier le garde-fou.
- Vérifier request_id et les compteurs stats.

14.2 SSO

- Enregistrer chaque callback exact chez le fournisseur.
- Tester une connexion avec un sso_subject pré-provisionné.
- Vérifier le refus d'un state modifié ou réutilisé.
- Tester une claim sub absente.
- Tester l'auto-liaison avec e-mail vérifié et domaine autorisé.
- Tester le refus d'un domaine non autorisé.
- Tester un compte reconnu mais sans accès à la boutique.
- Tester les parcours boutique et back-office séparément.

> **Critère de mise en production.** La recette est validée lorsque la synchronisation est reproductible, que les désactivations sont maîtrisées et que les connexions SSO ne créent ni ne relient de compte de manière ambiguë.

14.3 Informations à joindre au support

- request_id de la réponse API ou date et heure précise du parcours SSO ;
- réseau, boutique et code fournisseur concernés ;
- statut HTTP et code d'erreur, sans aucun secret ;
- external_id concerné et sujet SSO haché ou tronqué ;
- résultat attendu et résultat observé.

15. Dépannage

Symptôme	Cause probable	Vérification
`401 unauthorized`	Clé absente, erronée ou révoquée	Contrôler le secret et le réseau associé.
`413 payload_too_large`	JSON ou volume trop important	Scinder l'appel partial ; vérifier les limites.
`422 champ inconnu`	Propriété hors contrat	Comparer le JSON aux tableaux de référence.
`422 e-mail déjà utilisé`	Nouvel external_id en collision	Conserver l'external_id d'origine ou arbitrer côté métier.
`Full bloqué`	Plus de 30 % d'inactifs prévus	Contrôler l'exhaustivité et les status inactive.
`SSO boucle ou callback refusé`	redirect_uri non autorisée	Comparer caractère par caractère le callback enregistré.
`SSO compte non autorisé`	Aucun lien et auto-liaison impossible	Synchroniser sso_subject ou corriger les claims.
`SSO accès boutique refusé`	Habilitation absente	Vérifier rôle, agence, shop_codes et statut.
`Conflit de sujet SSO`	Sujet déjà lié à un autre compte	Ne pas réaffecter un sub ; corriger la source d'identité.

15.1 Contrôle rapide d'une intégration

1. Vérifier le domaine et le Content-Type.
2. Vérifier que la clé appartient au bon réseau.
3. Valider le JSON et supprimer tout champ non documenté.
4. Comparer external_id avec les envois précédents.
5. Contrôler les codes boutiques et l'agence de l'utilisateur.
6. Conserver puis rechercher le request_id.

Annexe A. Payload de référence

```
{
  "sync_mode": "full",
  "locations": [
    {
      "external_id": "AG-001",
      "name": "Agence Lyon",
      "address": "18 rue Exemple",
      "postal_code": "69002",
      "city": "Lyon",
      "phone": "+33 4 00 00 00 00",
      "email": "lyon@partenaire.fr",
      "status": "active",
      "shop_codes": ["LYON-PRO"]
    }
  ],
  "users": [
    {
      "external_id": "USR-10042",
      "sso_subject": "00u8abc123xyz",
      "sso_provider_code": "partenaire-pro",
      "firstname": "Camille",
      "lastname": "Martin",
      "email": "camille.martin@partenaire.fr",
      "role": "ROLE_LOCATION_MANAGER",
      "status": "active",
      "location_external_id": "AG-001",
      "shop_codes": ["LYON-PRO"]
    }
  ]
}
```

Ne pas copier tel quel en production. En mode full, ce document doit contenir l'intégralité des agences et utilisateurs attendus. Les objets actifs absents sont désactivés, sous réserve du garde-fou.

Annexe B. Glossaire

Terme	Définition
`external_id`	Identifiant stable d'une agence ou d'un utilisateur dans le système partenaire.
`sso_subject`	Identifiant externe immuable du compte chez le fournisseur OAuth, généralement sub.
`provider code`	Code court identifiant un fournisseur SSO dans un réseau.
`partial`	Synchronisation ciblée sans désactivation des objets absents.
`full`	Instantané exhaustif avec désactivation des objets actifs absents.
`request_id`	Identifiant unique de requête utilisé pour la corrélation et le support.
`userinfo`	Endpoint OAuth renvoyant les claims du compte connecté.
`claim`	Attribut JSON renvoyé par le fournisseur d'identité.

Fin du guide • Version 1.0 • Publication du 11 juin 2026