Spécification technique Keycloak — Intégration support Telegram Ceteris Prime

Destinataire : Équipe technique Keycloak **Émetteur** : Ceteris Tours / Ange Kouassi **Version** : 2.0 — Avril 2026 **Objet** : Développements à réaliser côté Keycloak pour activer le support Telegram de Ceteris Prime

---

1. Contexte

Votre équipe a déjà mis en place Keycloak pour gérer l'authentification et le 2FA des utilisateurs d'Estair Connect. Cette intégration fonctionne et n'est pas concernée par ce document.

Ceteris Prime ajoute maintenant un **second canal d'accès** au service : un bot Telegram `@CeterisprimeBot` géré par un orchestrateur n8n (système self-hosted, déjà en place).

Ce document spécifie uniquement les **ajouts à faire côté Keycloak** pour intégrer ce canal Telegram dans le cycle de vie utilisateur.

---

2. Vue d'ensemble du besoin

2.1 Le parcours visé

Un utilisateur déjà authentifié sur Estair Connect (avec son 2FA configuré) doit pouvoir activer son accès au bot Telegram en 3 étapes :

- Il reçoit un écran Keycloak lui proposant d'activer Telegram
- Il clique sur un bouton qui génère un token et envoie un email
- Il clique sur le lien dans son mail, ouvre Telegram, et valide avec son TOTP

Après activation, il peut utiliser le bot librement sans intervention de Keycloak.

2.2 Ce que Keycloak doit faire

Trois choses nouvelles :

- **Afficher un écran** de proposition d'activation Telegram (Required Action custom)
- **Envoyer un mail d'activation** avec un token unique
- **Exposer un endpoint API** pour que n8n valide le token + TOTP

2.3 Ce que Keycloak ne fait pas

Pour éviter toute confusion :

- Keycloak ne reçoit pas les messages Telegram (c'est n8n qui s'en charge)
- Keycloak n'est pas sollicité à chaque message Telegram (seulement pendant l'activation)
- Keycloak ne stocke pas l'historique des conversations Telegram

---

3. Synthèse des 3 éléments à livrer

#	Élément	Type	Effort estimé
1	Required Action custom "Activer Telegram" + template mail	Dev Java	2 jours
2	Endpoint custom `/validate-activation`	Dev Java	1 jour
3	Client OAuth2 pour n8n (service account)	Config	30 min

Total estimé : 3-4 jours de développement + 30 minutes de configuration.

---

4. Élément 1 — Required Action "Activer Telegram"

4.1 Principe

Une Required Action Keycloak custom qui s'affiche à l'utilisateur après sa configuration 2FA réussie. Elle propose d'activer l'accès au bot Telegram via un mail.

4.2 Déclenchement

- La Required Action s'active automatiquement **après** que l'utilisateur a configuré son 2FA

Google Authenticator pour la première fois

- Mécanisme recommandé : event listener sur l'événement Keycloak `UPDATE_TOTP`
- À la prochaine étape du login, l'utilisateur voit l'écran décrit ci-dessous

4.3 Écran affiché à l'utilisateur

Contenu attendu :

- **Titre** : « Activer votre support Telegram »
- **Texte** :
  - « Vous pouvez utiliser notre bot Telegram `@CeterisprimeBot` pour poser vos questions et obtenir de l'aide. »
  - « Pour activer cet accès, cliquez sur le bouton ci-dessous. Vous recevrez un email avec un lien d'activation. »
  - « Ouvrez cet email depuis votre téléphone (là où Telegram est installé) pour finaliser l'activation. »
- **Bouton principal** : « Recevoir mon lien d'activation Telegram »

4.4 Action au clic sur le bouton

Au clic, Keycloak doit :

- **Générer un token aléatoire cryptographique**
  - Format : 32 caractères hexadécimaux
  - Utiliser `SecureRandom` (Java) ou équivalent sécurisé
- **Enregistrer le token dans les attributs user**
  - `telegram_activation_token` = le token généré
  - `telegram_activation_expires_at` = maintenant + 15 minutes (timestamp epoch millisecondes)
- **Envoyer un email** à l'utilisateur avec le template décrit en 4.6
- **Afficher un écran de confirmation** :
  - « Un email vient de vous être envoyé à `[email_user]` . »
  - « Consultez votre boîte de réception sur votre téléphone pour finaliser l'activation. »

- Bouton : « Continuer vers Estair Connect »

4.5 Gestion de la Required Action dans le temps

La Required Action reste active tant que le token n'a pas été consommé.

Conséquences :

- Si l'utilisateur ferme son navigateur sans consulter son mail, au prochain login il reverra l'écran
- Dans ce cas, un nouveau token est généré (l'ancien est invalidé) et un nouveau mail est envoyé
- La Required Action est désactivée uniquement quand le token est validé via l'endpoint `/validate-activation` (voir élément 2)

Cette logique évite qu'un utilisateur se retrouve sans possibilité d'activer Telegram s'il perd son premier mail.

4.6 Template du mail d'activation

Fichier : `telegram-activation.ftl` **Format** : Freemarker **Emplacement** : `themes/ceteris/email/html/telegram-activation.ftl` + version texte

Variables du template :

- `${user.firstAttribute('name')}` — nom de l'utilisateur
- `${token}` — token d'activation
- `${expiresInMinutes}` — durée avant expiration (valeur : 15)

Sujet du mail : « Activez votre support Telegram Ceteris Prime »

Contenu HTML suggéré :

```
<html>
<body style="font-family: Arial, sans-serif; max-width: 600px; margin: 20px auto;
  <h2 style="color: #1A3A6E;">Bonjour ${user.firstAttribute('name')!},</h2>

  <p>Votre compte Estair Connect est configuré.
  Pour finaliser l'activation de votre support Telegram,
  cliquez sur le bouton ci-dessous depuis votre téléphone :</p>

  <p style="text-align: center; margin: 30px 0;">
    <a href="https://t.me/CeterisprimeBot?start=${token}"
       style="background-color: #1E8FE1; color: white;
```

```
              padding: 14px 28px; text-decoration: none;
              border-radius: 6px; display: inline-block;">
      Activer le support Telegram
    </a>
  </p>

  <p style="color: #666; font-size: 13px;">
    Ce lien est valable <strong>${expiresInMinutes} minutes</strong>.
  </p>

  <p style="color: #666; font-size: 13px;">
    Si le bouton ne fonctionne pas, copiez-collez ce lien
    dans votre navigateur :<br>
    <span style="word-break: break-all;">
      https://t.me/CeterisprimeBot?start=${token}
    </span>
  </p>

  <hr style="margin-top: 40px; border: 0; border-top: 1px solid #eee;">
  <p style="color: #999; font-size: 11px;">
    Ceteris Prime<br>
    Vous recevez ce mail car vous avez terminé la configuration
    de votre compte Ceteris Prime. Si vous n'êtes pas à l'origine
    de cette action, contactez support@ceterisservices.com.
  </p>
</body>
</html>
```

Version texte brut (pour clients mail sans HTML) :

```
Bonjour ${user.firstAttribute('name')!},

Votre compte Estair Connect est configuré. Pour finaliser
l'activation de votre support Telegram, cliquez sur le lien
ci-dessous depuis votre téléphone :

https://t.me/CeterisprimeBot?start=${token}

Ce lien est valable ${expiresInMinutes} minutes.

---
Ceteris Prime
support@ceterisservices.com
```

4.7 Attributs utilisateur custom nécessaires

Les utilisateurs doivent pouvoir stocker les attributs suivants (à ajouter s'ils n'existent pas déjà) :

- `telegram_activation_token` (string) — token temporaire
- `telegram_activation_expires_at` (string ou long) — timestamp d'expiration
- `telegram_user_id` (string) — rempli après activation réussie

4.8 Livrables attendus pour l'élément 1

- Extension Java implémentant la Required Action custom
- Event listener activant automatiquement la Required Action après configuration TOTP
- Templates de mail (HTML + texte) intégrés dans un thème Keycloak custom
- Documentation de déploiement (comment installer l'extension)

---

5. Élément 2 — Endpoint `/validate-activation`

5.1 Principe

Un endpoint REST custom dans Keycloak qui permet à n8n (le bot Telegram) de valider en une seule opération :

- Le token d'activation que l'utilisateur a reçu par mail
- Le code TOTP que l'utilisateur saisit dans Telegram

Si les deux sont valides, Keycloak enregistre le `telegram_user_id` dans le profil de l'utilisateur et renvoie ses informations à n8n.

5.2 URL

```
POST /realms/ceteris/ceteris/validate-activation
```

5.3 Authentification

L'endpoint doit être protégé par OAuth2 :

- Header requis : `Authorization: Bearer <access_token>`
- Le token doit provenir du client `n8n-ceteris` (voir élément 3)

- Le token doit avoir le rôle/scope `ceteris-activation-validator`
- Retourner `401` si authentification invalide
- Retourner `403` si le client n'a pas le bon rôle

5.4 Requête

Body JSON :

```
{
  "token": "a3f5b2c1e8d9...",
  "totp_code": "456789",
  "telegram_user_id": "987654321"
}
```

Champs :

- `token` (string, requis) — le token reçu du bot Telegram via `/start <token>`
- `totp_code` (string, requis) — les 6 chiffres tapés par l'utilisateur dans Telegram
- `telegram_user_id` (string, requis) — l'ID du compte Telegram de l'utilisateur

5.5 Logique à implémenter

Dans l'ordre :

Étape 1 — Trouver l'utilisateur par le token

- Chercher un user qui a `telegram_activation_token = <token>` dans ses attributs
- Si introuvable → répondre `400` avec `reason: token_not_found`

Étape 2 — Vérifier la validité temporelle du token

- Lire `telegram_activation_expires_at` de l'user
- Si `expires_at < now` → répondre `400` avec `reason: token_expired`

Étape 3 — Vérifier que le telegram_user_id n'est pas déjà rattaché

- Chercher si un autre user (différent de celui trouvé) a `telegram_user_id = <telegram_user_id>`
- Si oui → répondre `400` avec `reason: telegram_already_linked`

Étape 4 — Valider le code TOTP

- Utiliser le SPI Keycloak pour récupérer le secret TOTP de l'user
- Calculer le TOTP attendu au moment présent (avec tolérance ±1 fenêtre de 30 secondes)
- Comparer avec le code fourni
- Si invalide → répondre `400` avec `reason: invalid_totp`

Étape 5 — Gestion du compteur d'échecs TOTP

- Maintenir un compteur d'échecs TOTP par token (peut être stocké comme attribut user temporaire : `totp_attempts`)
- Après 3 échecs : invalider le token (effacer `telegram_activation_token`) et répondre `reason: max_attempts_exceeded`
- Si la validation réussit, effacer le compteur

Étape 6 — En cas de succès

- Enregistrer `telegram_user_id` dans l'attribut user
- Effacer `telegram_activation_token` et `telegram_activation_expires_at` (token consommé)
- **Désactiver la Required Action "Activer Telegram"** pour cet user
- Répondre `200` avec les infos user

5.6 Réponses attendues

Réponse 200 — Succès

```
{
  "success": true,
  "user_id": "12345-abcde-67890",
  "email": "agent@example.com",
  "name": "Kouamé Assi",
  "role": "responsable",
  "compte_id": "recXYZ123"
}
```

Réponse 400 — Erreur de validation

```
{
  "success": false,
```

```
  "reason": "invalid_totp | token_not_found | token_expired | telegram_already_li
  "attempts_remaining": 2,
  "display_message": "Code incorrect. Il vous reste 2 essais."
}
```

Les `display_message` doivent être en **français**, prêts à afficher à l'utilisateur sans transformation :

Reason	Display message
`token_not_found`	« Ce lien n'est pas valide. Retournez dans Estair Connect pour générer un nouveau lien. »
`token_expired`	« Ce lien a expiré. Retournez dans Estair Connect pour générer un nouveau lien. »
`telegram_already_linked`	« Ce compte Telegram est déjà rattaché à une autre agence. »
`invalid_totp`	« Code incorrect. Il vous reste {attempts_remaining} essais. »
`max_attempts_exceeded`	« Trop de tentatives. Retournez dans Estair Connect pour générer un nouveau lien. »

Réponse 401 — Non authentifié

```
{
  "error": "unauthorized",
  "error_description": "Missing or invalid bearer token"
}
```

Réponse 403 — Permission refusée

```
{
  "error": "forbidden",
  "error_description": "Client does not have required role"
}
```

5.7 Livrables attendus pour l'élément 2

- Extension Java implémentant l'endpoint
- Tests unitaires couvrant les 6 étapes de logique

- Documentation de l'API (OpenAPI/Swagger si possible)

---

6. Élément 3 — Client OAuth2 pour n8n

6.1 Principe

Créer un client Keycloak dédié à n8n pour qu'il puisse s'authentifier auprès de Keycloak de manière sécurisée et appeler l'endpoint `/validate-activation`.

6.2 Paramètres du client

- **Client ID** : `n8n-ceteris`
- **Client type** : `Confidential` (avec client secret)
- **Access Type** : `Service Account` activé
- **Authorization Grants activés** :
  - `Client credentials` (pour s'authentifier de serveur à serveur)
- **Authorization Grants désactivés** :
  - Direct Access Grants
  - Standard Flow
  - Implicit Flow
- **Redirect URIs** : vide (pas utilisé pour service account)
- **Web Origins** : vide

6.3 Rôles et scopes du client

Le service account de ce client doit avoir accès à :

- Rôle custom `ceteris-activation-validator` (permet d'appeler l'endpoint `/validate-activation`)

6.4 Livrables attendus

À fournir à Ceteris de manière sécurisée (pas par email en clair) :

- `client_id`
- `client_secret`

- URL de l'endpoint de token : `https://<votre-keycloak>/realms/ceteris/protocol/openid-connect/token`

---

7. Points transverses

7.1 Sécurité

- **Tous les secrets** (client secrets, tokens) doivent être stockés dans un coffre sécurisé
- **Aucun log en clair** des tokens d'activation, codes TOTP ou secrets en production
- **TLS obligatoire** sur toutes les URLs Keycloak (`https://` uniquement)
- Considérer activer un **rate limit** sur l'endpoint `/validate-activation` (suggestion : 10 requêtes/minute par client)

7.2 Logs et observabilité

Les événements suivants doivent être loggués :

- Activation Telegram réussie
- Échecs TOTP répétés
- Tokens expirés utilisés
- Tentatives d'accès non authentifié à l'endpoint
- Envoi de mail d'activation (succès/échec)

7.3 Tests d'intégration avec n8n

Une fois les 3 éléments livrés, prévoir une session de tests conjointe avec l'équipe Ceteris :

- Test de l'authentification client credentials (n8n obtient un access token)
- Test d'appel à `/validate-activation` avec différents cas :
  - Token valide + TOTP correct → succès
  - Token valide + TOTP incorrect → erreur avec attempts_remaining
  - Token expiré → erreur
  - Token inexistant → erreur
  - TOTP échoué 3 fois → token invalidé

- Test du flow complet de bout en bout avec un utilisateur test (configuration 2FA → écran activation → mail → bot Telegram → validation)

---

8. Synthèse des livrables

Livrable	Format
Extension JAR déployée	Fichier `.jar` installé sur le serveur Keycloak
Thème email custom	Dossier de thème dans Keycloak
Client `n8n-ceteris`	`client_id` + `client_secret` (transmis par canal sécurisé)
Documentation API	Spec OpenAPI de l'endpoint `/validate-activation`
Documentation de déploiement	Markdown ou PDF expliquant comment installer l'extension
Rapport de tests	Résultats des tests unitaires et d'intégration

---

9. Calendrier proposé

Semaine	Activité
Semaine 1	Création du client `n8n-ceteris` + début dev Required Action
Semaine 2	Fin Required Action + dev endpoint `/validate-activation`
Semaine 3	Tests internes + tests d'intégration avec l'équipe Ceteris
Semaine 4	Corrections + déploiement staging + validation + déploiement production

---

10. Points à clarifier avant démarrage

Avant de démarrer le développement, les informations suivantes sont nécessaires :

- Votre équipe a-t-elle déjà développé des extensions Keycloak custom (Required Actions, endpoints REST, event listeners) ?

- Quelle version de Keycloak est déployée ? La version impacte les APIs SPI disponibles
- Existe-t-il un environnement de dev/staging déjà disponible, ou à créer ?
- Quels sont les outils de monitoring et de logs en place ?
- Qui est le point de contact technique pour cette intégration ?

---

11. Contacts

- **Contact Ceteris** : Ange Kouassi — Directeur Général
- **Canal de communication à définir** : Slack, Teams, mail, visio
- **Cadence de sync proposée** : point hebdomadaire de 30-45 min pendant les 4 semaines

---

Fin du document