heyalink

Grands comptes

GUIDE DU DÉVELOPPEUR

Se raccorder à l'API heyalink

Publier vos offres d'emploi et vos formations depuis votre logiciel.
À transmettre à la personne qui fera le branchement.

VOTRE PREMIER APPEL · CURL

```
curl https://heyalink.com/api/emploi/v1/moi \
  -H "Authorization: Bearer $HEYALINK_CLE"
```

API EMPLOI	`https://heyalink.com/api/emploi/v1`
API FORMATION	`https://heyalink.com/api/formation/v1`
SPÉCIFICATION	`https://heyalink.com/api/emploi/v1/openapi.json` OpenAPI 3.1, à importer dans Postman, Insomnia ou un générateur de client.

Sommaire

8 ÉTAPES, DANS L'ORDRE DU BRANCHEMENT

01 **Obtenir sa clé** 3
Une clé par logiciel, ses droits, le bac à sable et la preuve de domaine.

02 **Faire un premier appel** 5
Vérifier la clé, puis essayer une publication à blanc.

03 **Publier une offre** 6
Votre référence, un PUT : l'offre est en ligne. Les champs, le lot, la synchronisation.

04 **Publier une formation** 11
Les mêmes gestes pour vos formations, et votre catalogue LHEO tel quel.

05 **Recevoir les webhooks signés** 17
Être prévenu de chaque changement, et vérifier la signature.

06 **Brancher un agent d'IA** 19
Le serveur MCP, avec la même clé et les mêmes droits.

07 **Respecter les limites** 20
Le débit, la reprise sans risque, la lecture sans retéléchargement.

08 **Comprendre les erreurs** 21
Le format RFC 9457, chaque code et ce qu'il faut faire, le journal.

ANNEXES

A **Toutes les routes** 23

B **Versions et journal des changements** 26

> **La documentation en ligne fait foi.** Ce guide en est tiré le jour de son édition. La version à jour, avec un bouton « Essayer » sur chaque exemple : heyalink.com/developpeurs.

01 Obtenir sa clé

Une clé par logiciel, ses droits, le bac à sable et la preuve de domaine.

Vos deux adresses

Posez-les une fois dans votre terminal : chaque commande du guide les reprend.

VOS ADRESSES · BASH

```
export HEYALINK_EMPLOI="https://heyalink.com/api/emploi/v1"
export HEYALINK_FORMATION="https://heyalink.com/api/formation/v1"
```

La clé

- Elle commence par `hl_emploi_live_` et compte 63 caractères. Ce préfixe lisible permet aux outils de détection de secrets (GitHub, GitGuardian) de la repérer si elle est oubliée dans un dépôt.
- Elle va dans l'en-tête de chaque appel : `Authorization: Bearer hl_emploi_live_VOTRECLE`. Jamais dans une adresse, un dépôt de code ou une page web publique.
- Elle appartient à **une** page et à **cette API seule** : elle ne lit ni ne touche les offres d'une autre page, et une clé de l'API Formation reçoit 403 ici (comme celle-ci sur l'API Formation).
- Elle ne marche que tant que la page a la **formule Grands comptes** : sans elle, chaque appel répond 403.
- Elle ne fait que ce que ses droits permettent : `offres:lecture`, `offres:ecriture` (publier permet aussi de lire).
- Facultatif, à sa création : une **durée de vie** (30 jours, 90 jours, 1 an) et une **liste d'adresses IP** autorisées (par exemple `203.0.113.0/24`) ; un appel venu d'ailleurs répond 403.
- Une page peut avoir jusqu'à dix clés actives sur l'API Emploi : une par logiciel, pour pouvoir en couper une sans toucher les autres. **Révoquer** une clé la coupe aussitôt.
- **Remplacer** une clé (rotation) en crée une nouvelle aux mêmes réglages, durée de vie comprise ; l'ancienne marche encore 7 jours, le temps de la changer dans votre logiciel, sans coupure. Si la clé a fuité, choisissez le **remplacement d'urgence** : l'ancienne est coupée tout de suite, avec ses jetons OAuth.
- Une clé d'essai (`hl_emploi_test_`) a sa propre limite d'appels, plus basse que celle d'une vraie clé (`GET /moi` la donne).

Le bac à sable

Une **clé d'essai** commence par `hl_emploi_test_`. Elle donne **exactement les mêmes réponses, validations et erreurs** qu'une vraie clé, mais ses offres vivent à part : elles ne sont jamais publiées ni vues des membres. Vous pouvez les créer, les lister, les modifier, les archiver et les supprimer comme de vraies offres ; elles sont effacées 7 jours après leur dernière modification. Chaque réponse porte `"bac_a_sable": true` et l'en-tête `Heyalink-Bac-A-Sable: oui`. Vos webhooks de test sont prévenus aussi, avec `"bac_a_sable": true` dans `donnees`. Branchez votre logiciel avec elle, puis passez à une clé `hl_emploi_live_`.

Le **journal de test** (`GET /journal` avec la clé d'essai, ou l'écran API Emploi) garde chaque appel avec sa requête et sa réponse complètes, secrets masqués.

La validation à blanc

Avec une vraie clé, ajoutez `?dry_run=1` à n'importe quelle écriture (ou l'en-tête `Heyalink-Simulation: 1`) : tout est vérifié (clé, droits, champs, référence) et la réponse est celle d'une vraie écriture, avec `"simulation": true`. **Rien n'est écrit.**

REQUÊTE · CURL

```
curl -X PUT "$HEYALINK_EMPLOI/offres/CC-2026-014?dry_run=1" \
  -H "Authorization: Bearer hl_emploi_live_VOTRECLE" \
  -H "Content-Type: application/json" \
  -d '{
        "titre": "Chef de chantier gros œuvre",
        "description": "<p>Deux équipes.</p>",
        "lieu": "Toulouse"
      }'
```

OAuth 2.0

Si votre logiciel préfère OAuth, échangez la clé contre un **jeton d'une heure** (octroi « client credentials », RFC 6749) : `client_id` est l'identifiant client de la clé (`hlc_…`, affiché sur l'écran API Emploi), `client_secret` la clé elle-même. `scope` restreint les droits du jeton (facultatif), pris parmi `offres:lecture` et `offres:ecriture`.

REQUÊTE · CURL

```
curl -X POST "$HEYALINK_EMPLOI/oauth/token" \
  -u "hlc_VOTREIDENTIFIANT:hl_emploi_live_VOTRECLE" \
  -d grant_type=client_credentials \
  -d scope="offres:ecriture"
```

RÉPONSE · JSON

```
{
  "access_token": "hl_at_…",
  "token_type": "Bearer",
  "expires_in": 3600,
  "scope": "offres:ecriture"
}
```

Le jeton s'emploie ensuite comme une clé : `Authorization: Bearer hl_at_…`. Il ne vaut que pour l'API Emploi, et tombe avec la clé (révocation, expiration).

Les droits d'une clé

DROIT	API	CE QU'IL PERMET
`offres:lecture`	API Emploi	Lire vos offres
`offres:ecriture`	API Emploi	Publier vos offres
`sous_traitance:lecture`	API Emploi	Lire vos demandes de sous-traitance et leurs candidatures
`sous_traitance:ecriture`	API Emploi	Publier vos demandes de sous-traitance
`formations:lecture`	API Formation	Lire vos formations
`formations:ecriture`	API Formation	Publier vos formations

Prouver votre domaine

Une publication pour de bon (et sa validation à blanc) demande que votre page ait **prouvé qu'elle possède le domaine de son site**. Un SIRET vérifié dit que l'entreprise existe, pas que c'est bien elle qui publie : la preuve de domaine protège vos candidats contre une page qui usurperait votre nom. Le bac à sable, lui, reste ouvert sans preuve. Les domaines prouvés sont ceux de la page : ils valent pour l'API Emploi comme pour l'API Formation.

1. Déclarez le domaine : `POST /domaines` avec `{"domaine": "votre-site.fr"}`, ou sur l'écran API Emploi. La réponse donne un **jeton** propre au domaine.
2. Posez-le sur votre site, au choix : la balise `<meta name="heyalink-verification" content="JETON">` dans la page d'accueil, ou un fichier `https://votre-site.fr/.well-known/heyalink.txt` qui contient le jeton.
3. Vérifiez : `POST /domaines/votre-site.fr/verification`. `domaine.prouve` passe à `true`.

Le domaine prouvé vaut aussi pour ses sous-domaines (`carrieres.votre-site.fr`). Le lien de candidature de vos offres (`lien_candidature`) doit mener à un domaine prouvé de votre page : sinon l'offre est refusée en 422, sur le champ. Une page carrières ou un flux déjà prouvés comme source valent preuve de leur domaine. La preuve est revue chaque semaine : retirée de votre site, elle ne compte plus. Il n'y a pas d'autre voie : pas de validation à la main. `GET /domaines` liste vos domaines, `DELETE /domaines/votre-site.fr` en retire un.

02 Faire un premier appel

Vérifier la clé, puis essayer une publication à blanc.

Vérifiez-la

L'appel `moi` dit à quelle page appartient la clé, ses droits et vos limites.

REQUÊTE · CURL

```
curl $HEYALINK_EMPLOI/moi \
  -H "Authorization: Bearer $HEYALINK_CLE"
```

RÉPONSE · JSON

```
{
  "entreprise": {
    "nom": "Bâtisseurs de Garonne",
    "adresse": "https://heyalink.com/…"
  },
  "cle": {
    "nom": "Logiciel de recrutement", "api": "emploi", "debut": "hl_emploi_live_k3",
    "droits": ["offres:lecture", "offres:ecriture"], "creee_le": "2026-10-28T10:12:00+01:00"
  },
  "limites": { "appels_par_minute": 300, "annonces_par_lot": 100 },
  "success": true
}
```

Essayez à blanc

Avec `dry_run=1`, tout est vérifié et la réponse est celle d'une vraie écriture, avec `"simulation": true`. Rien n'est publié.

REQUÊTE · CURL

```
curl -X PUT "$HEYALINK_EMPLOI/offres/CC-2026-014?dry_run=1" \
  -H "Authorization: Bearer $HEYALINK_CLE" \
  -H "Content-Type: application/json" \
  -d @offre.json
```

03 Publier une offre

Votre référence, un PUT : l'offre est en ligne. Les champs, le lot, la synchronisation.

Les offres d'emploi

Les appels

APPEL		CE QU'IL FAIT	DROIT
GET	/offres	Lister vos offres d'emploi	offres:lecture
GET	/offres/{reference}	Lire une annonce et son état	offres:lecture
PUT	/offres/{reference}	Publier ou modifier (idempotent)	offres:ecriture
DELETE	/offres/{reference}	Supprimer (un PUT la ramène)	offres:ecriture
POST	/offres/{reference}/archivage	Archiver	offres:ecriture
POST	/offres/{reference}/pourvue	Marquer pourvue	offres:ecriture
POST	/offres/{reference}/fermer	Fermer (l'annonce ne reçoit plus de réponses)	offres:ecriture
POST	/offres/{reference}/rouvrir	Rouvrir une annonce fermée	offres:ecriture
GET	/offres/{reference}/apercu	La fiche telle qu'un membre la voit	offres:lecture
POST	/offres/{reference}/pieces	Joindre des photos et des PDF	offres:ecriture
PUT	/offres/{reference}/pieces/ordre	Ranger les photos	offres:ecriture
DELETE	/offres/{reference}/pieces/{piece}	Retirer une photo ou un PDF	offres:ecriture
POST	/offres/lot	Envoyer jusqu'à 100 annonces et retraits	offres:ecriture
POST	/offres/synchronisation	Synchronisation complète : ce qui manque est archivé	offres:ecriture

Les champs d'une offre

CHAMP	FORMAT	DESCRIPTION
titre OBLIGATOIRE	texte, 200 signes au plus	L'intitulé du poste. Exemple : `Chef de chantier gros œuvre`

CHAMP	FORMAT	DESCRIPTION
description OBLIGATOIRE	texte, 20 000 signes au plus	Le texte de l'offre ; le HTML est accepté : les balises sont retirées, le texte est gardé.
lieu OBLIGATOIRE	texte, 255 signes au plus	La ville ; vide, la ville de votre page. Exemple : `Toulouse (31)`
code_postal	5 chiffres	Il range l'offre dans son département. Exemple : `31000`
type_contrat	texte, 60 signes au plus	CDI, CDD, Intérim, Apprentissage, Alternance, Stage ou Saisonnier ; les mots anglais des sites d'emploi sont compris (permanent, temporary, internship…). Exemple : `CDI`
salaire	texte, 50 signes au plus	Le salaire en clair, tel que vous l'affichez.
metier	texte, 120 signes au plus	Le métier, en texte ; reconnu, il fait proposer l'offre aux bons membres. Exemple : `Chef de chantier`
secteur	texte, 120 signes au plus	Le secteur d'activité, en texte. Exemple : `Gros œuvre`
lien_candidature	adresse https://	L'adresse où postuler chez vous, sur un domaine prouvé de votre page ; absente, on postule sur Heyalink. Exemple : `https://www.votre-site.fr/offres/CC-2026-014`
date_expiration	date AAAA-MM-JJ	Ignoré depuis le 07/11/2026 (v1.10.0) : facultatif, accepté mais sans effet. Une offre n'a plus de date de fin : elle reste en ligne jusqu'à ce que vous l'archiviez. Exemple : `2026-12-31`
metier_code	code, 64 signes au plus	Le code d'un métier du référentiel de Heyalink. Inconnu : refusé (422). Sans lui, Heyalink déduit le métier du code ROME, de `metier` ou de l'intitulé. Exemple : `chef-de-chantier-batiment`
famille_code	code, 64 signes au plus	Le code d'une des 39 familles de métiers de Heyalink. Sans lui, la famille du métier. Exemple : `entreprise-generale`
code_rome	texte au motif `^[A-N]\d{4}$`	Le code ROME de France Travail (une lettre, quatre chiffres). Exemple : `F1202`
experience_mois	entier, de 0 à 600	L'expérience demandée, en mois ; 0 pour un débutant accepté.
temps_de_travail	un parmi : `plein`, `partiel`	Le temps de travail. Un `type_contrat` FULL_TIME ou PART_TIME le donne aussi. Exemple : `plein`
teletravail	un parmi : `non`, `partiel`, `total`	Le télétravail : non, partiel (hybride) ou total. Exemple : `non`
alternance	un parmi : `apprentissage`, `professionnalisation`	Le ou les contrats d'alternance. Exemple : `apprentissage`
salaire_min	nombre	Le bas de la fourchette de salaire brut, en euros, pour `salaire_periode`.
salaire_max	nombre	Le haut de la fourchette, au moins égal à `salaire_min`.
salaire_periode	un parmi : `heure`, `jour`, `semaine`, `mois`, `an`	L'unité de temps de `salaire_min` et `salaire_max` ; obligatoire avec l'un des deux. Exemple : `an`

CHAMP	FORMAT	DESCRIPTION
description_poste	texte, 10 000 signes au plus	v1.9.0 : le bloc « Le poste » de la fiche, en texte (une ligne par paragraphe, « - » en tête d'une puce). Facultatif ; avec les blocs, la fiche les montre à la place de `description`, qui reste exigée.
description_missions	texte, 10 000 signes au plus	v1.9.0 : le bloc « Les missions » de la fiche, même forme que `description_poste`.
description_profil	texte, 10 000 signes au plus	v1.9.0 : le bloc « Le profil » de la fiche, même forme que `description_poste`. Exemple : `- Trois ans d'expérience en gros œuvre - Lecture de plans`
description_avantages	texte, 10 000 signes au plus	v1.9.0 : le bloc « Les avantages » de la fiche, même forme que `description_poste`. Exemple : `- Véhicule de service - Panier repas`
duree_nombre	entier, de 1 à 999	v1.9.0 : la durée du contrat (CDD, intérim, saisonnier), avec `duree_unite`. La fiche l'écrit « Intérim, 18 mois ».
duree_unite	un parmi : `jours`, `semaines`, `mois`	v1.9.0 : l'unité de `duree_nombre` ; obligatoire avec lui. Exemple : `mois`

Les champs à partir de `metier_code` (v1.6.0) sont facultatifs et suivent des listes fermées : une valeur hors liste est refusée en 422, avec un exemple valide. Ils servent les filtres d'Emploi : **une offre sans la donnée sort du filtre correspondant**, elle n'en reçoit jamais une fausse.

REQUÊTE · CURL

```
curl -X PUT "$HEYALINK_EMPLOI/offres/CC-2026-014" \
  -H "Authorization: Bearer hl_emploi_live_VOTRECLE" \
  -H "Content-Type: application/json" \
  -d '{
        "titre": "Chef de chantier gros œuvre",
        "description": "<p>Vous dirigez deux équipes sur des chantiers de logements.</p>",
        "lieu": "Toulouse",
        "code_postal": "31000",
        "metier": "Chef de chantier",
        "type_contrat": "CDI",
        "salaire": "38 000 € par an"
      }'
```

RÉPONSE · JSON

```
{
  "offre": {
    "reference": "CC-2026-014",
    "titre": "Chef de chantier gros œuvre",
    "departement": "31",
    "metier": "Chef de chantier",
    "etat": "en_ligne",
    "adresse": "https://heyalink.com/emplois/chef-de-chantier-gros-oeuvre-12",
    "source": "api"
  },
  "avertissements": [],
  "success": true
}
```

La réponse dit aussi ce qui s'est passé, dans `resultat` : `creee`, `modifiee`, `restauree` (revenue en ligne), `inchangee` (rien n'a changé, rien n'est réécrit) ou `doublon` (le même contenu est déjà en ligne sous une autre référence : rien n'est

créé).

Une offre peut attendre avant de paraître (Grands comptes). Quand le plafond de facture du mois est atteint, ou qu'une facture reste impayée quatorze jours après son échéance, l'offre est bien enregistrée mais reste hors ligne : `etat` vaut `en_attente` et `attente` dit pourquoi (`motif` : `plafond` ou `impaye`), depuis quand, et `parait_au_plus_tard_le` (au plafond, le premier jour du mois suivant ; plus tôt si vous relevez votre plafond ; sur un impayé, au paiement). Elle se relit par sa référence. Le webhook `annonce.en_attente` vous prévient, puis `annonce.publiee` quand elle paraît. Pour toute autre offre, `attente` vaut `null`.

Les avertissements ne bloquent jamais : ils disent ce qui manque pour bien ranger l'offre (un métier que Heyalink ne reconnaît pas, avec les plus proches ; un lieu sans département). Corrigez-les quand vous pouvez : l'offre sera mieux proposée.

Les signalements (`signalements`, depuis la v1.6.0) ne bloquent jamais non plus. Chacun a un `code`, un `champ` et un `message` :

CODE	SENS
`champ_ignore`	vous avez envoyé un champ que l'API ne connaît pas : il n'est ni publié ni refusé (vingt au plus sont nommés)
`champ_recommande_absent`	un champ manque, et l'offre sortira du filtre qu'il remplit (expérience, salaire, contrat…)
`valeur_deduite`	vous n'avez pas envoyé ce champ : l'algorithme de Heyalink l'a déduit, `origine` dit d'où (`code_rome`, `metier`, `titre`, `metier_code`, `code_naf`, `type_contrat`)
`incoherence`	deux champs qui ne vont pas ensemble

Une valeur déduite n'est jamais présentée comme la vôtre : l'offre la liste dans `champs_deduits`. Envoyez le champ pour en décider vous-même. Le grand secteur (`secteur`) se lit depuis la famille, au moment de la réponse.

Relire une offre, et la liste de vos offres :

REQUÊTE · CURL

```
curl "$HEYALINK_EMPLOI/offres/CC-2026-014" \
  -H "Authorization: Bearer hl_emploi_live_VOTRECLE"
```

REQUÊTE · CURL

```
curl "$HEYALINK_EMPLOI/offres?limite=100" \
  -H "Authorization: Bearer hl_emploi_live_VOTRECLE"
```

Le poste est pourvu : l'offre sort du site, archivée « pourvue ».

REQUÊTE · CURL

```
curl -X POST "$HEYALINK_EMPLOI/offres/CC-2026-014/pourvue" \
  -H "Authorization: Bearer hl_emploi_live_VOTRECLE"
```

Une offre retirée par la modération de Heyalink ne revient jamais, ni par l'API (422 sur `reference`), ni par une source : écrivez à l'équipe si vous pensez à une erreur.

Supprimée ou fermée sur Heyalink (depuis la v1.11.0)

Un gérant de votre page peut fermer ou supprimer l'offre sur Heyalink.

- **Supprimée** : elle ne revient jamais. Renvoyer la même référence (`PUT`, envoi par lot) répond `409`, `code annonce_supprimee`, avec la date (`supprimee_le`) et ce texte : « Cette annonce a été supprimée sur heyalink le … par un gérant de la page. Pour la republier, utilisez une nouvelle référence. » Un flux (page carrières, logiciel) la saute et le note dans son bilan (`supprimees_sur_le_site`) et dans le journal.
- **Fermée** : renvoyée, elle se met à jour mais reste fermée (état `fermee`) ; seul un gérant la rouvre, ou vous par `POST /offres/{reference}/rouvrir`.

Envoyer par lot

Pour synchroniser toutes vos offres, envoyez jusqu'à 100 offres et 100 retraits par appel. Chaque offre a sa propre réponse : une offre refusée n'empêche pas les autres.

REQUÊTE · CURL

```
curl -X POST "$HEYALINK_EMPLOI/offres/lot" \
  -H "Authorization: Bearer hl_emploi_live_VOTRECLE" \
  -H "Content-Type: application/json" \
  -d '{
        "offres": [
          {
            "reference": "CC-2026-014",
            "titre": "Chef de chantier",
            "description": "…",
            "lieu": "Toulouse"
          },
          {"reference": "MAC-2026-002", "titre": "Maçon", "description": "…", "lieu": "Albi"}
        ],
        "retirer": ["CC-2025-099"]
      }'
```

La réponse donne un `bilan` (créées, modifiées, retirées, refusées) et, pour chaque offre, son statut (201, 200 ou 422) avec le message d'erreur s'il y en a un.

La liste complète. `POST /offres/synchronisation` reçoit toutes vos références en ligne : chaque offre de l'API qui n'y est pas est archivée. Une liste vide n'archive rien, sauf avec `"confirmer_liste_vide": true` : une erreur d'envoi ne vide pas votre page.

REQUÊTE · CURL

```
curl -X POST "$HEYALINK_EMPLOI/offres/synchronisation" \
  -H "Authorization: Bearer hl_emploi_live_VOTRECLE" \
  -H "Content-Type: application/json" \
  -d '{"references": ["CC-2026-014", "MAC-2026-002"]}'
```

04 Publier une formation

Les mêmes gestes pour vos formations, et votre catalogue LHEO tel quel.

Les formations

Les appels

APPEL		CE QU'IL FAIT	DROIT
GET	**/formations**	Lister vos formations	`formations:lecture`
GET	**/formations/{reference}**	Lire une annonce et son état	`formations:lecture`
PUT	**/formations/{reference}**	Publier ou modifier (idempotent)	`formations:ecriture`
DELETE	**/formations/{reference}**	Supprimer (un PUT la ramène)	`formations:ecriture`
POST	**/formations/{reference}/archivage**	Archiver	`formations:ecriture`
POST	**/formations/{reference}/pourvue**	Marquer pourvue	`formations:ecriture`
POST	**/formations/{reference}/fermer**	Fermer (l'annonce ne reçoit plus de réponses)	`formations:ecriture`
POST	**/formations/{reference}/rouvrir**	Rouvrir une annonce fermée	`formations:ecriture`
GET	**/formations/{reference}/apercu**	La fiche telle qu'un membre la voit	`formations:lecture`
POST	**/formations/{reference}/pieces**	Joindre des photos et des PDF	`formations:ecriture`
PUT	**/formations/{reference}/pieces/ordre**	Ranger les photos	`formations:ecriture`
DELETE	**/formations/{reference}/pieces/{piece}**	Retirer une photo ou un PDF	`formations:ecriture`
POST	**/formations/lot**	Envoyer jusqu'à 100 annonces et retraits	`formations:ecriture`
POST	**/formations/synchronisation**	Synchronisation complète : ce qui manque est archivé	`formations:ecriture`
POST	**/formations/lheo**	Envoyer un catalogue LHEO tel quel	`formations:ecriture`

Les champs d'une formation

CHAMP	FORMAT	DESCRIPTION
titre OBLIGATOIRE	texte, 200 signes au plus	L'intitulé de la formation. Exemple : `CACES R482 catégorie B1`
presentation OBLIGATOIRE	texte, 15 000 signes au plus	Le contenu de la formation ; le HTML est accepté : les balises sont retirées, le texte est gardé.
lien OBLIGATOIRE	adresse https://	L'adresse de la fiche ou de l'inscription chez vous. Exemple : `https://www.votre-organisme.fr/formations/caces-r482-b1`

CHAMP	FORMAT	DESCRIPTION
organisme	texte, 255 signes au plus	L'organisme ; absent, le nom de votre page. Exemple : `Centre de formation de la construction`
mode	texte, 60 signes au plus	presentiel, distanciel ou mixte. Exemple : `presentiel`
lieu	texte, 255 signes au plus	La ville. Exemple : `Blagnac`
code_postal	5 chiffres	Il range la formation dans son département. Exemple : `31700`
duree_heures	entier, de 1 à 99 999	La durée, en heures.
debut	date AAAA-MM-JJ	La date de la prochaine session. Exemple : `2026-11-16`
pour_qui	texte, 255 signes au plus	Le public visé. Exemple : `Conducteurs d'engins débutants`
niveau_d_entree	texte, 255 signes au plus	Le niveau demandé, en texte libre (voir aussi `niveau_entree`). Exemple : `Sans niveau requis`
prerequis	texte, 255 signes au plus	Les prérequis. Exemple : `Avoir 18 ans`
objectifs	texte, 255 signes au plus	Les objectifs de la formation. Exemple : `Conduire une pelle de moins de 6 tonnes en sécurité`
infos_pratiques	texte, 500 signes au plus	Prix, financement, horaires : ce qu'il faut savoir avant de s'inscrire.
metier	texte, 120 signes au plus	Le métier visé, en texte. Exemple : `Conducteur d'engins de chantier`
metier_code	code, 64 signes au plus	Le code d'un métier du référentiel de Heyalink. Inconnu : refusé (422). Sans lui, Heyalink déduit le métier du code ROME, de `metier` ou de l'intitulé. Exemple : `conducteur-d-engins-de-chantier`
famille_code	code, 64 signes au plus	Le code d'une des 39 familles de métiers de Heyalink. Sans lui, la famille du métier. Exemple : `terrassement-demolition`
code_rome	texte au motif `^[A-N]\d{4}$`	Le code ROME de France Travail (une lettre, quatre chiffres). Exemple : `F1302`
financements	liste, parmi : `cpf`, `opco`, `france_travail`, `region`, `employeur`	Les financements possibles. « cpf » suppose un code RNCP ou RS (sinon un signalement `incoherence`).
type_certification	un parmi : `diplome`, `habilitation`, `attestation`	Ce que prépare la formation : diplome (diplôme ou titre), habilitation (habilitation, CACES ou certificat), attestation. Exemple : `habilitation`
code_rncp	texte au motif `^RNCP\d{1,6}$`	Le code RNCP de la certification préparée (France Compétences), par exemple RNCP34567. Exemple : `RNCP34567`
code_rs	texte au motif `^RS\d{1,6}$`	Le code RS (Répertoire spécifique, France Compétences), par exemple RS5080. Exemple : `RS5080`
alternance	liste, parmi : `apprentissage`, `professionnalisation`	Le ou les contrats d'alternance.
niveau_entree	un parmi : `aucun`, `cap`, `bac`, `bac2`, `bac3`, `bac5`, `bac8`	Le niveau demandé à l'entrée : aucun, cap, bac, bac2, bac3, bac5, bac8. Exemple : `aucun`

CHAMP	FORMAT	DESCRIPTION
prix	nombre	Le prix total TTC par stagiaire, en euros. Toujours TTC : la fiche l'affiche ainsi, « 1 500 € TTC ».
certification	texte, 190 signes au plus	L'intitulé de la certification préparée, par exemple « CAP Couvreur » ou « CACES® R482 catégorie B1 ». Avec `code_rncp` ou `code_rs`, la fiche montre la carte « Ce que vous obtenez » ; sans code, la certification est affichée comme déclarée par l'organisme (v1.8.0). Exemple : `CACES® R482 catégorie B1`
diplome	texte, 40 signes au plus	Le sigle du diplôme d'État préparé : CAP, BP, BTS, Bac pro, BUT… Il fait écrire « Diplôme national » sur la fiche (v1.8.0). Exemple : `CAP`
niveau_certification	entier, de 1 à 8	Le niveau européen de la certification (1 à 8), seulement avec un `code_rncp`. Le niveau du répertoire national passe devant (v1.8.0).
attestation	vrai ou faux	Vrai si la formation se termine par une attestation de fin de formation ; affiché seulement s'il est vrai (v1.8.0).
photo_url	adresse https://	L'adresse https d'une photo de la formation (vos ateliers, une session) : JPEG, PNG, GIF ou WebP, 5 Mo au plus, servie par une adresse publique. Heyalink la recopie et la remet au propre (2 000 px de large au plus). Illisible : refusée (422). Envoyée à null : la photo est retirée ; absente : la photo reste (v1.8.0). Elle passe ensuite par la vérification des images de Heyalink, comme une photo de profil : tant qu'elle est vérifiée, `photo` rend la précédente ; refusée, elle n'est jamais publiée (07/11/2026). Exemple : `https://www.votre-organisme.fr/images/plateau-engins.jpg`
photo_base64	texte, 7 000 100 signes au plus	La même photo, envoyée en base64 (avec ou sans en-tête `data:`), à la place de `photo_url` (v1.8.0). Exemple : `…`

Les champs à partir de `metier` (v1.6.0) sont facultatifs et suivent des listes fermées : une valeur hors liste est refusée en 422, avec un exemple valide. Ils servent les filtres de Formation : **une formation sans la donnée sort du filtre correspondant** (financement, certification, alternance, niveau). « cpf » sans code RNCP ni RS est accepté, avec un signalement `incoherence`.

La fiche de la formation (v1.8.0) se remplit avec ces champs, comme une formation saisie à la main sur Heyalink. En tête, la carte « Ce que vous obtenez » : `certification`, le code (`code_rncp` ou `code_rs`), le niveau sur 8 pour un RNCP (celui du répertoire national passe devant `niveau_certification`) et la nature du titre (« Diplôme national » avec un `diplome` d'État, « Certification » pour un code RS ; un code que le répertoire national ne connaît pas est affiché comme déclaré par l'organisme). Sans certification, la carte devient « Ce que vous saurez faire » et montre vos `objectifs`, puis la durée, puis l'attestation si `attestation` est vrai. Sans l'un ni l'autre, pas de carte. La famille de métier est calculée par Heyalink (le titre d'abord, puis `famille_code`) et rendue en lecture dans `famille`. **La photo** est facultative : Heyalink la lit à l'adresse donnée (adresse publique seulement), la vérifie et la remet au propre (2 000 px de large au plus). Illisible, la formation est refusée (422, champ `photo_url`). Une photo absente de la requête reste ; `photo_url` envoyé à `null` la retire. Une nouvelle photo passe ensuite par la vérification des images de Heyalink, comme une photo de profil : tant qu'elle est vérifiée, `photo` rend la précédente ; refusée, elle n'est jamais publiée. Le prix (`prix`) est toujours TTC : la fiche l'affiche « 1 500 € TTC ».

REQUÊTE · CURL

```
curl -X PUT "$HEYALINK_FORMATION/formations/CACES-R482-B1" \
  -H "Authorization: Bearer hl_formation_live_VOTRECLE" \
  -H "Content-Type: application/json" \
  -d '{
        "titre": "CACES R482 catégorie B1",
        "presentation": "Conduite en sécurité des pelles hydrauliques de moins de 6 tonnes.",
        "lien": "https://www.votre-organisme.fr/formations/caces-r482-b1",
        "mode": "presentiel",
        "lieu": "Blagnac",
        "code_postal": "31700",
        "duree_heures": 21,
        "debut": "2026-11-16"
      }'
```

RÉPONSE · JSON

```
{
  "formation": {
    "reference": "CACES-R482-B1",
    "titre": "CACES R482 catégorie B1",
    "mode": "presentiel",
    "lieu": "Blagnac",
    "etat": "en_ligne",
    "adresse": "https://heyalink.com/formations/caces-r482-categorie-b1-8",
    "source": "api"
  },
  "avertissements": [],
  "success": true
}
```

La réponse dit aussi ce qui s'est passé, dans `resultat` : `creee`, `modifiee`, `restauree` (revenue en ligne), `inchangee` (rien n'a changé, rien n'est réécrit) ou `doublon` (le même contenu est déjà en ligne sous une autre référence : rien n'est créé). **Une formation peut attendre avant de paraître** (Grands comptes) : au plafond de facture du mois, ou sur une facture impayée quatorze jours après son échéance, elle est enregistrée mais reste hors ligne. `etat` vaut alors `en_attente` et `attente` dit pourquoi (`motif` : `plafond` ou `impaye`), depuis quand, et `parait_au_plus_tard_le`. Le webhook `annonce.en_attente` vous prévient, puis `annonce.publiee` quand elle paraît. Pour toute autre formation, `attente` vaut `null`. **Les avertissements** ne bloquent jamais : ils disent ce qui manque pour bien ranger la formation (un lieu sans département, par exemple).

Les signalements (`signalements`, depuis la v1.6.0) ne bloquent jamais non plus : `code` `champ_ignore` (un champ que l'API ne connaît pas, ni publié ni refusé), `champ_recommande_absent` (un champ qui manque, et la formation sortira du filtre qu'il remplit), `valeur_deduite` (l'algorithme de Heyalink l'a déduit à votre place ; `origine` dit d'où : `metier`, `code_rome`, `titre`, `code_rncp`, `code_rs`…) ou `incoherence`. Une valeur déduite est listée dans `champs_deduits`, jamais présentée comme la vôtre.

Dans un catalogue LHEO (2.3), ces données se lisent sans saisie : `domaine-formation/code-ROME`, `organisme-financeur/code-financeur` (2 Conseil régional, 4 France Travail, 5 Entreprise, 16 OPCO), `certification/code-RNCP` et `code-RS`, `code-CPF` (éligible au CPF), `certifiante`, `code-niveau-entree`, `prix-total-TTC` et l'alternance des sessions (`session/cycle/alternance/type-alternance`).

Relire une formation, et la liste de vos formations :

REQUÊTE · CURL

```
curl "$HEYALINK_FORMATION/formations/CACES-R482-B1" \
  -H "Authorization: Bearer hl_formation_live_VOTRECLE"
```

REQUÊTE · CURL

```
curl "$HEYALINK_FORMATION/formations?limite=100" \
  -H "Authorization: Bearer hl_formation_live_VOTRECLE"
```

La session est complète : la formation sort du site, archivée « pourvue ».

REQUÊTE · CURL

```
curl -X POST "$HEYALINK_FORMATION/formations/CACES-R482-B1/pourvue" \
  -H "Authorization: Bearer hl_formation_live_VOTRECLE"
```

Votre catalogue LHEO s'envoie aussi tel quel, en corps de requête (5 Mo au plus). Avec `?synchronisation=complete`, les formations absentes du catalogue sont archivées.

REQUÊTE · CURL

```
curl -X POST "$HEYALINK_FORMATION/formations/lheo?synchronisation=complete" \
  -H "Authorization: Bearer hl_formation_live_VOTRECLE" \
  -H "Content-Type: application/xml" \
  --data-binary @catalogue-lheo.xml
```

Une formation masquée par la modération de Heyalink ne se republie pas (422) et ne se retire pas par l'API : écrivez à l'équipe si vous pensez à une erreur.

Supprimée ou fermée sur Heyalink (depuis la v1.11.0)

Un gérant de votre page peut fermer ou supprimer la formation sur Heyalink.

- **Supprimée** : elle ne revient jamais. Renvoyer la même référence (`PUT`, envoi par lot) répond `409`, `code annonce_supprimee`, avec la date (`supprimee_le`) et ce texte : « Cette annonce a été supprimée sur heyalink le … par un gérant de la page. Pour la republier, utilisez une nouvelle référence. » Un flux (page carrières, logiciel) la saute et le note dans son bilan (`supprimees_sur_le_site`) et dans le journal.
- **Fermée** : renvoyée, elle se met à jour mais reste fermée (état `fermee`) ; seul un gérant la rouvre, ou vous par `POST /formations/{reference}/rouvrir`.

Envoyer par lot

Pour synchroniser un catalogue entier, envoyez jusqu'à 100 formations et 100 retraits par appel. Chaque formation a sa propre réponse : une formation refusée n'empêche pas les autres.

REQUÊTE · CURL

```
curl -X POST "$HEYALINK_FORMATION/formations/lot" \
  -H "Authorization: Bearer hl_formation_live_VOTRECLE" \
  -H "Content-Type: application/json" \
  -d '{
        "formations": [
          {
            "reference": "CACES-R482-B1",
            "titre": "CACES R482 catégorie B1",
            "presentation": "Pelles de moins de 6 tonnes.",
            "lien": "https://www.votre-organisme.fr/formations/caces-r482-b1"
          },
          {
            "reference": "SST-2026-01",
            "titre": "Sauveteur secouriste du travail",
            "presentation": "Deux jours pour savoir porter secours.",
            "lien": "https://www.votre-organisme.fr/formations/sst"
          }
        ],
        "retirer": ["HAB-2025-099"]
      }'
```

La réponse donne un `bilan` (créées, modifiées, retirées, refusées) et, pour chaque formation, son statut (201, 200 ou 422) avec le message d'erreur s'il y en a un.

La liste complète. `POST /formations/synchronisation` reçoit toutes vos références en ligne : chaque formation de l'API qui n'y est pas est archivée. Une liste vide n'archive rien, sauf avec `"confirmer_liste_vide": true` : une erreur d'envoi ne vide pas votre page.

REQUÊTE · CURL

```
curl -X POST "$HEYALINK_FORMATION/formations/synchronisation" \
  -H "Authorization: Bearer hl_formation_live_VOTRECLE" \
  -H "Content-Type: application/json" \
  -d '{"references": ["CACES-R482-B1", "SST-2026-01"]}'
```

Sans développement : votre catalogue

Sur l'écran API Formation, collez l'adresse `https://` de votre catalogue : au format **LHEO** (celui des Carif-Oref) ou en **JSON** (une liste de formations, ou un objet `{"formations": […]}`, avec les champs de la table plus haut). Heyalink le relit et tient vos formations à jour.

D'abord, prouvez que la source est la vôtre. Pour que personne ne publie vos formations à votre place, Heyalink ne lit une source qu'une fois prouvé qu'elle vous appartient. L'écran API Formation vous donne un jeton propre à la source, à poser d'une de ces façons, puis cliquez « Vérifier » :

- sur votre site, une balise dans l'en-tête de votre page d'accueil : `<meta name="heyalink-verification" content="VOTRE-JETON">` ;
- ou un fichier texte `https://votre-site/.well-known/heyalink.txt` qui contient le jeton.

Si vous ne pouvez pas poser le jeton, branchez-vous par l'API (plus haut) : `POST /formations/lheo` envoie le même catalogue avec votre clé.

Laissez le jeton en place : Heyalink le revérifie chaque semaine. S'il disparaît, la source n'est plus lue et vous êtes prévenu par courriel.

Chaque lecture compare tout : une formation nouvelle est créée, une formation modifiée mise à jour, une formation disparue de votre catalogue archivée (jamais effacée : si elle revient, elle revient avec son adresse).

Prudence en cas de panne : une lecture en erreur ne retire rien ; un catalogue qui ne rend soudain plus aucune formation n'est vidé qu'au passage suivant, s'il est toujours vide. Au bout de trois erreurs de suite, ou dès qu'il se vide, vous êtes prévenu par courriel.

Fraîcheur : Heyalink relit votre catalogue toutes les 10 minutes au plus vite, plus rarement s'il ne change pas (jusqu'à 6 heures), en requêtes conditionnelles (`ETag`, `Last-Modified`) pour ne télécharger que ce qui change, et s'annonce comme `HeyalinkBot`. « Synchroniser maintenant » se demande une fois toutes les cinq minutes au plus.

Tout de suite, si vous voulez : appelez l'**adresse de mise à jour immédiate** de votre source (écran API Formation, « Mise à jour immédiate ») à chaque changement, Heyalink relit votre catalogue dans la minute. Si votre catalogue annonce un hub **WebSub**, Heyalink s'y abonne seul, avec un secret propre au hub (signatures `sha1`, `sha256`, `sha384` ou `sha512` acceptées).

REQUÊTE · CURL

```
curl -X POST "$HEYALINK_FORMATION/sources/12/relire/VOTRE-SIGNATURE"
```

05 Recevoir les webhooks signés

Être prévenu de chaque changement, et vérifier la signature.

Les webhooks

Plutôt que d'interroger Heyalink, faites-vous prévenir. Sur l'écran API Emploi, ajoutez une adresse `https://` de votre site (cinq au plus) et choisissez ses événements. Une adresse de l'API Emploi n'est prévenue que des offres d'emploi : elle n'apprend rien des formations.

ÉVÉNEMENT	QUAND
`annonce.publiee`	l'offre est en ligne (créée, revenue d'une archive, ou sortie d'attente)
`annonce.en_attente`	l'offre est enregistrée mais attend hors ligne : plafond de facture atteint ou facture impayée (Grands comptes)
`annonce.refusee`	la modération de Heyalink l'a masquée ou retirée
`annonce.expiree`	sa source (flux, page carrières) la dit échue ; depuis le 07/11/2026, une offre envoyée par l'API n'expire plus
`annonce.archivee`	elle est archivée
`annonce.pourvue`	elle est marquée pourvue
`annonce.fermee`	un gérant l'a fermée (sur Heyalink ou par `…/fermer`) ; elle se rouvre
`annonce.rouverte`	une annonce fermée est de nouveau en ligne
`annonce.supprimee`	un gérant l'a supprimée sur Heyalink, définitivement

ÉVÉNEMENT	QUAND
`annonce.piece_refusee`	la modération a refusé une photo ou un PDF de l'annonce ; `donnees.piece` porte `id`, `type` (`photo` ou `document`) et `nom`
`ping`	l'essai envoyé depuis l'écran API Emploi

Une adresse peut être une **adresse de test** : elle ne reçoit que les événements de vos offres de test (clé `hl_emploi_test_`), et une vraie adresse que ceux de vos vraies offres, jamais les uns chez les autres. Un événement de test porte `"bac_a_sable": true` dans son corps signé et l'en-tête `Heyalink-Bac-A-Sable: oui`. Si l'une de vos adresses accumule trop de livraisons en attente (panne de votre côté), les nouvelles ne sont plus préparées jusqu'à ce qu'elle réponde.

Heyalink envoie un `POST` en JSON :

EXEMPLE · JSON

```
{
  "id": "evt_8f3k2m1q9x0a7b6c5d4e",
  "evenement": "annonce.refusee",
  "cree_le": "2026-10-19T14:02:11+02:00",
  "bac_a_sable": false,
  "donnees": {
    "nature": "offre",
    "reference": "CC-2026-014",
    "titre": "Chef de chantier",
    "motif": null,
    "adresse": "https://heyalink.com/emplois/chef-de-chantier-12"
  }
}
```

Vérifiez la signature avant de croire un appel. L'en-tête `Heyalink-Signature: t=<horodatage>,v1=<signature>` porte un HMAC-SHA256, calculé avec le secret de l'adresse (affiché une seule fois à sa création) sur `<horodatage>.<corps brut>`. Refusez un horodatage de plus de cinq minutes : un appel rejoué plus tard sera refusé.

CODE · PHP

```
[$t, $v1] = sscanf($_SERVER['HTTP_HEYALINK_SIGNATURE'], 't=%d,v1=%s');
$corps = file_get_contents('php://input');
$attendu = hash_hmac('sha256', $t.'.'.$corps, getenv('HEYALINK_SECRET'));
if (! hash_equals($attendu, (string) $v1) || abs(time() - $t) > 300) {
    http_response_code(400);
    exit;
}
```

CODE · JAVASCRIPT

```
// corpsBrut : le corps reçu, tel quel (avant tout JSON.parse).
const [, t, v1] = req.get('Heyalink-Signature').match(/t=(\d+),v1=([a-f0-9]+)/);
const attendu = crypto.createHmac('sha256', process.env.HEYALINK_SECRET)
  .update(`${t}.${corpsBrut}`)
  .digest('hex');
const valide = attendu.length === v1.length
  && crypto.timingSafeEqual(Buffer.from(attendu), Buffer.from(v1))
  && Math.abs(Date.now() / 1000 - Number(t)) <= 300;
```

CODE · PYTHON

```python
# entete : l'en-tête Heyalink-Signature ; corps_brut : le corps reçu, en octets.
t, v1 = (partie.split('=', 1)[1] for partie in entete.split(','))
attendu = hmac.new(os.environ['HEYALINK_SECRET'].encode(),
                   t.encode() + b'.' + corps_brut, hashlib.sha256).hexdigest()
valide = hmac.compare_digest(attendu, v1) and abs(time.time() - int(t)) <= 300
```

Répondez par un statut 2xx dans les 10 secondes. Sinon, Heyalink réessaie après 1, 5, 30 minutes, puis 2 h, 6 h et 24 h ; `Heyalink-Livraison` identifie la livraison, pour ne pas traiter deux fois la même. Une adresse en échec 20 fois de suite est désactivée.

06 Brancher un agent d'IA

Le serveur MCP, avec la même clé et les mêmes droits.

Le serveur MCP

Un agent d'IA (Claude, ChatGPT, ou votre propre agent) peut publier et tenir à jour vos offres par le **Model Context Protocol** : `POST https://heyalink.com/api/emploi/v1/mcp`, transport « Streamable HTTP », avec la même clé et les mêmes droits que l'API (`Authorization: Bearer hl_emploi_live_…`). Une clé d'essai ne publie rien.

Outils, ceux de l'API Emploi seulement : `offres_lister`, `offre_publier`, `offre_archiver`, `offre_pourvue`, `offre_supprimer`.

EXEMPLE · JSON

```json
{
  "mcpServers": {
    "heyalink-emploi": {
      "type": "http",
      "url": "https://heyalink.com/api/emploi/v1/mcp",
      "headers": { "Authorization": "Bearer hl_emploi_live_VOTRECLE" }
    }
  }
}
```

Pour l'API Formation

Outils, ceux de l'API Formation seulement : `formations_lister`, `formation_publier`, `formation_archiver`, `formation_pourvue`, `formation_supprimer`.

EXEMPLE · JSON

```json
{
  "mcpServers": {
    "heyalink-formation": {
      "type": "http",
      "url": "https://heyalink.com/api/formation/v1/mcp",
      "headers": { "Authorization": "Bearer hl_formation_live_VOTRECLE" }
    }
  }
}
```

07 Respecter les limites

Le débit, la reprise sans risque, la lecture sans retéléchargement.

Adresse et format

Adresse de base	`https://heyalink.com/api/emploi/v1`
Format	JSON en UTF-8 (`Content-Type: application/json`), réponses en JSON
Description complète	OpenAPI 3.1 : `https://heyalink.com/api/emploi/v1/openapi.json`, à importer dans Postman, Insomnia ou un générateur de client
Limite	300 appels par minute et par clé, annoncés sur chaque réponse

Limite d'appels. Chaque réponse porte les en-têtes `RateLimit-Policy` et `RateLimit` (appels restants, secondes avant la remise à zéro). Au-delà, la réponse est 429 avec `Retry-After` : attendez ce nombre de secondes, ou regroupez vos offres avec `/lot`.

Rejouer sans risque. Ajoutez `Idempotency-Key: <un identifiant unique>` à une écriture : rejouée avec la même valeur dans les 24 heures (après une coupure réseau, par exemple), elle rend la même réponse sans rien refaire, avec `Idempotent-Replayed: true`. La même valeur pour une autre requête répond 422.

Lire sans retélécharger. Chaque lecture porte un `ETag` ; renvoyez-le en `If-None-Match` : si rien n'a changé, la réponse est 304, sans corps.

Langue. Les messages sont en français ; envoyez `Accept-Language: en` pour les avoir en anglais.

État de l'API. `GET https://heyalink.com/api/emploi/v1/sante`, public et sans clé, donne l'état et la version :

REQUÊTE · CURL

```
curl $HEYALINK_EMPLOI/sante
```

Chaque offre est désignée par **votre référence** (le numéro de l'offre chez vous) : lettres, chiffres, `-`, `_`, `.` et `:`, 191 caractères au plus. C'est elle qui rend les appels sûrs à rejouer : envoyer deux fois la même référence modifie l'offre, ne la double jamais.

08 Comprendre les erreurs

Le format RFC 9457, chaque code et ce qu'il faut faire, le journal.

Les erreurs

Une erreur répond au format standard **RFC 9457** (`application/problem+json`), avec un message en français lisible tel quel :

EXEMPLE · JSON

```
{
  "type": "https://heyalink.com/developpeurs/api/emploi#erreur-422",
  "title": "Annonce refusée",
  "status": 422,
  "detail": "Le champ titre est obligatoire.",
  "instance": "/api/emploi/v1/offres/CC-2026-014",
  "errors": { "titre": ["Le champ titre est obligatoire."] },
  "violations": [
    {
      "champ": "titre",
      "regle": "required",
      "message": "Le champ titre est obligatoire.",
      "exemple": "Chef de chantier gros œuvre"
    }
  ]
}
```

`violations` dit, pour chaque champ refusé, la règle qui a échoué et une valeur valide : de quoi corriger sans chercher.

CODE	SIGNIFICATION	QUE FAIRE
200, 201	modifiée, créée	rien
204	retirée	rien
304	rien n'a changé depuis votre `ETag`	garder ce que vous avez
400	requête mal formée (`Idempotency-Key` trop longue, par exemple)	lire `detail`
401	clé absente, inconnue, révoquée ou expirée	vérifier l'en-tête, recréer une clé
403	clé de l'API Formation, page sans la formule Grands comptes, droit manquant, adresse IP non autorisée, ou domaine pas encore prouvé	lire `detail`
404	référence inconnue (lecture seulement)	vérifier la référence
409	la même `Idempotency-Key` est encore en cours	attendre sa réponse
422	offre refusée : champ manquant ou trop long, lien bloqué, offre retirée par la modération, ou `Idempotency-Key` déjà prise	lire `errors`, champ par champ
429	trop d'appels	attendre `Retry-After` secondes, ou passer par `/lot`

Les champs d'une erreur

CHAMP	FORMAT	DESCRIPTION
type OBLIGATOIRE	adresse https://	L'adresse de l'explication de cette erreur dans la documentation.
title OBLIGATOIRE	texte	Le nom de l'erreur, en clair.
status OBLIGATOIRE	entier	Le code HTTP de la réponse.
detail OBLIGATOIRE	texte	Ce qui ne va pas et quoi corriger, lisible tel quel.
instance OBLIGATOIRE	texte	Le chemin de l'appel refusé.
errors	objet	Les messages, champ par champ.
violations	liste	Pour chaque champ refusé : la règle qui a échoué, le message et une valeur valide.

Le journal

`GET /journal` rend les 100 derniers appels et lectures de sources de l'API Emploi, avec leur statut et leur message : une clé ne lit que le journal de son API. Il se lit avec une clé qui a le droit `offres:ecriture`. Le même journal se lit sur l'écran API Emploi. Il est gardé 90 jours ; le journal de test (requêtes et réponses des clés d'essai, secrets masqués) l'est 7 jours.

REQUÊTE · CURL

```
curl "$HEYALINK_EMPLOI/journal" \
  -H "Authorization: Bearer hl_emploi_live_VOTRECLE"
```

Retirer une offre se rejoue sans risque : la réponse est 204, même si elle était déjà retirée.

REQUÊTE · CURL

```
curl -X DELETE "$HEYALINK_EMPLOI/offres/MAC-2026-002" \
  -H "Authorization: Bearer hl_emploi_live_VOTRECLE"
```

A

ANNEXE

Toutes les routes

Tirées des spécifications OpenAPI 3.1 des deux API.

API Emploi

`https://heyalink.com/api/emploi/v1`

APPEL		CE QU'IL FAIT	DROIT
GET	/offres	Lister vos offres d'emploi	offres:lecture
GET	/offres/{reference}	Lire une annonce et son état	offres:lecture
PUT	/offres/{reference}	Publier ou modifier (idempotent)	offres:ecriture
DELETE	/offres/{reference}	Supprimer (un PUT la ramène)	offres:ecriture
POST	/offres/{reference}/archivage	Archiver	offres:ecriture
POST	/offres/{reference}/pourvue	Marquer pourvue	offres:ecriture
POST	/offres/{reference}/fermer	Fermer (l'annonce ne reçoit plus de réponses)	offres:ecriture
POST	/offres/{reference}/rouvrir	Rouvrir une annonce fermée	offres:ecriture
GET	/offres/{reference}/apercu	La fiche telle qu'un membre la voit	offres:lecture
POST	/offres/{reference}/pieces	Joindre des photos et des PDF	offres:ecriture
PUT	/offres/{reference}/pieces/ordre	Ranger les photos	offres:ecriture
DELETE	/offres/{reference}/pieces/{piece}	Retirer une photo ou un PDF	offres:ecriture
POST	/offres/lot	Envoyer jusqu'à 100 annonces et retraits	offres:ecriture
POST	/offres/synchronisation	Synchronisation complète : ce qui manque est archivé	offres:ecriture
GET	/sous-traitance	Lister les demandes de sous-traitance de votre page	sous_traitance:lecture
POST	/sous-traitance	Publier une demande de sous-traitance au nom de votre page	sous_traitance:ecriture
GET	/sous-traitance/{demande}	Lire une demande et son état	sous_traitance:lecture
PATCH	/sous-traitance/{demande}	Modifier une demande, bloc par bloc	sous_traitance:ecriture
DELETE	/sous-traitance/{demande}	Supprimer, définitivement	sous_traitance:ecriture
POST	/sous-traitance/{demande}/fermer	Fermer (la demande ne reçoit plus de candidatures)	sous_traitance:ecriture
POST	/sous-traitance/{demande}/rouvrir	Rouvrir une demande fermée	sous_traitance:ecriture
GET	/sous-traitance/{demande}/candidatures	Les candidatures reçues, en lecture	sous_traitance:lecture
GET	/moi	Vérifier la clé : sa page, son API, ses droits	toute clé
GET	/journal	Les 100 derniers appels et lectures (le journal de test pour une clé d'essai)	toute clé

APPEL		CE QU'IL FAIT	DROIT
POST	/mcp	Serveur MCP (Model Context Protocol, JSON-RPC 2.0) pour les agents d'IA	toute clé
GET	/domaines	Les domaines de votre page et leur preuve	toute clé
POST	/domaines	Ajouter un domaine à prouver : le jeton à poser sur votre site	toute clé
POST	/domaines/{domaine}/verification	Chercher la preuve sur votre site (balise meta ou /.well-known/heyalink.txt)	toute clé
DELETE	/domaines/{domaine}	Retirer un domaine	toute clé
POST	/oauth/token	OAuth 2.0, client credentials : un jeton d'une heure	sans clé
GET	/openapi.json	Cette spécification	sans clé
GET	/sante	L'état de l'API (public)	sans clé
GET	/services	Les services de la page : brouillons, en ligne, fermés	toute clé
POST	/services	Créer un service en brouillon (type et title facultatifs : ils remplissent la tête)	toute clé
GET	/services/{id}	Un service : son état, ses blocs, ce qui manque avant de publier	toute clé
DELETE	/services/{id}	Supprimer le service (définitif ; webhook annonce.supprimee)	toute clé
PATCH	/services/{id}/blocs/{bloc}	Enregistrer un bloc : seuls ses champs changent	toute clé
POST	/services/{id}/blocs/{bloc}	Enregistrer un bloc en multipart (photos, PDF) : comme PATCH	toute clé
GET	/services/{id}/apercu	Voir comme un membre : ce que la fiche montre	toute clé
POST	/services/{id}/publication	Publier le service (tous les blocs obligatoires remplis, dans la limite de la formule)	toute clé
POST	/services/{id}/fermeture	Fermer le service : il sort des listes et du plafond (webhook annonce.fermee)	toute clé
POST	/services/{id}/reouverture	Rouvrir le service, dans la limite de la formule (webhook annonce.rouverte)	toute clé

API Formation

`https://heyalink.com/api/formation/v1`

Les routes du compte (/moi, /journal, /mcp, /domaines, /oauth/token, /openapi.json, /sante) existent aussi à l'adresse de l'API Formation, avec une clé de l'API Formation.

APPEL		CE QU'IL FAIT	DROIT
GET	/formations	Lister vos formations	formations:lecture
GET	/formations/{reference}	Lire une annonce et son état	formations:lecture
PUT	/formations/{reference}	Publier ou modifier (idempotent)	formations:ecriture
DELETE	/formations/{reference}	Supprimer (un PUT la ramène)	formations:ecriture
POST	/formations/{reference}/archivage	Archiver	formations:ecriture
POST	/formations/{reference}/pourvue	Marquer pourvue	formations:ecriture
POST	/formations/{reference}/fermer	Fermer (l'annonce ne reçoit plus de réponses)	formations:ecriture
POST	/formations/{reference}/rouvrir	Rouvrir une annonce fermée	formations:ecriture
GET	/formations/{reference}/apercu	La fiche telle qu'un membre la voit	formations:lecture
POST	/formations/{reference}/pieces	Joindre des photos et des PDF	formations:ecriture
PUT	/formations/{reference}/pieces/ordre	Ranger les photos	formations:ecriture
DELETE	/formations/{reference}/pieces/{piece}	Retirer une photo ou un PDF	formations:ecriture
POST	/formations/lot	Envoyer jusqu'à 100 annonces et retraits	formations:ecriture
POST	/formations/synchronisation	Synchronisation complète : ce qui manque est archivé	formations:ecriture
POST	/formations/lheo	Envoyer un catalogue LHEO tel quel	formations:ecriture

B

ANNEXE

Versions et journal des changements

Ce qui a changé, version par version.

Un contrat stable

L'API Emploi est en **v1** (`https://heyalink.com/api/emploi/v1`). Ce contrat est stable : la v1 n'évolue que de façon compatible (un champ ou un appel de plus, jamais un champ retiré ni un sens changé). Une rupture passerait en v2, à une autre adresse, annoncée à l'avance. La description complète, à jour, est la spécification OpenAPI (`https://heyalink.com/api/emploi/v1/openapi.json`), qui fait foi.

VERSION	DATE	CHANGEMENTS

1.13.0	08/11/2026	Changement compatible (rien de retiré ni de renommé). Les demandes de sous-traitance de la page : `GET` et `POST /sous-traitance`, `GET`, `PATCH` et `DELETE /sous-traitance/{demande}`, `POST …/fermer` et `…/rouvrir`, `GET …/candidatures` (en lecture). Deux droits de plus, `sous_traitance:lecture` et `sous_traitance:ecriture` (une clé d'avant ne les a pas : en créer une nouvelle). Cinq webhooks : `sous_traitance.publiee`, `.fermee`, `.rouverte`, `.supprimee`, `.candidature_recue`
1.12.0	07/11/2026	Changement compatible (rien de retiré ni de renommé). Photos et PDF de l'annonce : `media[]` dans la lecture (`id`, `type`, `nom`, `url`, `taille`, `ordre`, `statut` : `verification`, `publiee` ou `refusee`), `POST …/pieces`, `PUT …/pieces/ordre`, `DELETE …/pieces/{piece}`, et le webhook `annonce.piece_refusee`. Chaque pièce est lue par la modération avant d'être montrée aux membres. Une annonce importée n'en porte pas
1.11.0	07/11/2026	Changement compatible (rien de retiré ni de renommé). Fermer et rouvrir : `POST /offres/{reference}/fermer` et `…/rouvrir` ; l'aperçu tel qu'un membre le voit : `GET /offres/{reference}/apercu`. Deux états de plus (`fermee`, `retiree`) et trois webhooks (`annonce.fermee`, `annonce.rouverte`, `annonce.supprimee`). Une annonce supprimée sur Heyalink par un gérant ne revient jamais : la même référence répond `409 annonce_supprimee` ; une annonce fermée renvoyée reste fermée. Les services de la page, bloc par bloc : `GET /services`, `POST /services` (un brouillon), `GET /services/{id}`, `PATCH /services/{id}/blocs/{bloc}` (seuls les champs du bloc changent ; photos et PDF en multipart), `GET /services/{id}/apercu`, `POST /services/{id}/publication`, `POST /services/{id}/fermeture`, `POST /services/{id}/reouverture`, `DELETE /services/{id}`. La lecture suit le droit de lecture de la clé, l'écriture son droit d'écriture. Les trois mêmes webhooks disent aussi le geste de la page sur un service (nature `service`, référence : l'identifiant du service), et `annonce.publiee` sa publication
1.10.0	07/11/2026	Plus de date de fin : `date_expiration` reste accepté mais est ignoré (aucune erreur, aucun effet) et vaut toujours `null` dans les réponses ; une offre reste en ligne jusqu'à son archivage (`DELETE` ou `statut`). Une date reçue avant ne ferme plus l'offre. Le webhook `annonce.expiree` ne vise plus que les offres reprises d'une source (flux, page carrières) que la source dit échues. Aucun champ retiré ni renommé
1.9.0	07/11/2026	Changement compatible (rien de retiré ni de renommé, aucun sens changé). La fiche de l'offre : quatre champs facultatifs `description_poste`, `description_missions`, `description_profil`, `description_avantages` (les blocs de la fiche ; `description` reste exigée) et `duree_nombre`, `duree_unite` (la durée du contrat). Le texte `salaire` reste accepté ; sans `salaire_min` ni `salaire_max`, il donne les trois valeurs du salaire seulement quand il suit une forme sûre (« Mensuel de 2100.0 Euros à 2500.0 Euros », « 2 200 à 2 500 € par mois »), sinon il s'affiche tel quel. Mise à jour d'une offre déjà publiée (`PUT` sur la même référence) : un champ absent ne change plus rien, un champ obligatoire envoyé vide est refusé (422) ; une création exige toujours `titre`, `description` et `lieu`. Une offre publiée par l'API s'affiche dans la fiche exactement comme une offre saisie sur heyalink
1.8.0	06/11/2026	Aucun changement pour l'API Emploi : la version suit celle de l'API Formation (la fiche Formation)
1.7.0	30/10/2026	Documentation seule : elle dit la règle d'accès en vigueur depuis le 25/10/2026 : l'API Emploi est comprise dans la formule Grands comptes ; l'API elle-même ne change pas
1.7.0	29/10/2026	Dans la réponse, `secteur_de_recherche` : le secteur (un des dix) sous lequel la offre se trouve dans les filtres de recherche de Heyalink, lu depuis la famille ; aucun champ retiré, aucun sens changé
1.6.0	28/10/2026	Données riches, toutes facultatives : `metier_code`, `famille_code`, `code_rome`, `experience_mois`, `temps_de_travail`, `teletravail`, `alternance`, `salaire_min`, `salaire_max`, `salaire_periode` (listes fermées, 422 hors liste) ; dans la réponse, ces champs plus `secteur` et `champs_deduits` ; bloc `signalements` (champs ignorés, champs recommandés absents, valeurs déduites, incohérences) dans chaque publication et chaque résultat d'un lot ; le temps de travail, l'expérience, le télétravail et le salaire en nombres sont lus aussi dans les pages carrières balisées et les flux ; aucun champ retiré, aucun sens changé

1.5.0	26/10/2026	État `en_attente` et champ `attente` (motif `plafond` ou `impaye`, date de parution au plus tard) pour les annonces d'un Grand compte retenues par le plafond de facture ou un impayé ; webhook `annonce.en_attente` ; une annonce en attente se relit par sa référence et figure dans la liste
1.4.0	21/10/2026	Deux API distinctes, l'API Emploi (`/api/emploi/v1`) et l'API Formation (`/api/formation/v1`), chacune avec ses clés (`hl_emploi_live_`, `hl_emploi_test_`), sa spécification, ses collections, ses webhooks, son journal et ses outils MCP ; une clé de l'autre API reçoit 403 ; `cle.api` dans `GET /moi` ; l'ancienne adresse `/api/entreprises/v1` et les anciennes clés restent servies jusqu'à leur remplacement
1.3.0	19/10/2026	Domaines prouvés (`/domaines`) : exigés pour publier pour de bon, et les liens des annonces doivent y mener ; adresses de webhook de test, champ `bac_a_sable` et en-tête `Heyalink-Bac-A-Sable` ; la validation à blanc vaut aussi pour le serveur MCP ; remplacement d'urgence d'une clé ; limite d'appels propre aux clés d'essai ; le journal demande un droit de publication ; le secret OAuth ne passe plus dans l'adresse
1.2.0	19/10/2026	Bac à sable complet (annonces de test listables, effacées après 7 jours), validation à blanc (`dry_run`), `violations` dans les erreurs, messages en anglais (`Accept-Language`), route `sante`, journal de test, données d'exemple, collections Postman et Bruno, « Tester ma source »
1.1.0	19/10/2026	Clés `hl_live_` et `hl_test_`, droits fins, durée de vie, adresses IP, rotation, OAuth 2.0, `Idempotency-Key`, en-têtes `RateLimit`, erreurs RFC 9457, curseur, `ETag`, webhooks signés, OpenAPI 3.1, serveur MCP, preuve de propriété des sources
1.0.0	19/10/2026	Offres d'emploi et formations par référence, lot, synchronisation, LHEO, sources lues sans développement