Guide de migration

De la v2. à la v3.

Ce guide liste l'ensemble des changements effectués entre la version 3 de l'API Entreprise et la version 2, et vous livre les éléments nécessaires pour effectuer la migration.
Les évolutions présentées ici ont été guidées par trois objectifs :

- ➔ assurer une meilleure sécurité de la donnée des fournisseurs ;
- ➔ normaliser les formats pour faciliter la compréhension et l'industrialisation ;
- ➔ clarifier, documenter les réponses et les rendre actionnables par vos logiciels.

I. Évolutions générales

Jeton d'accès à paramétrer dans le header

Avec la V3 :
Le jeton est à paramétrer uniquement dans le header de l'appel.

Avant :
Le jeton JWT pouvait être un paramètre de l'URL d'appel (query parameter).

Pourquoi ?

- Respecter les standards de sécurité
- Garantir que le token ne sera pas utilisé dans un navigateur.

Comment ?
Utiliser un client REST API pour tester les API pendant le développement.
Des clients sont disponibles gratuitement. API Entreprise utilise pour ses propres tests le client Insomnia. Le plus connu sur le marché est Postman.
Une fois le client installé, vous pouvez directement intégrer notre fichier Swagger/OpenAPI dedans.

Votre numéro de SIRET obligatoire dans le récipient

Avec la V3 :
Le paramètre obligatoire 'recipient' de l'URL d'appel devra obligatoirement être complété par **votre** numéro de SIRET.

Avant :
Ce paramètre obligatoire n'était pas contraint en termes de syntaxe.

Pourquoi ?

- Pour garantir la traçabilité de l'appel jusqu'au bénéficiaire ayant obtenu l'habilitation à appeler l'API Entreprise et respecter nos engagements auprès des fournisseurs de données.
- Nous avions trop d'utilisateurs inscrivant le numéro de SIRET ou RNA de l'entreprise/association recherchée.

Cas particulier ⚠️ : *Vous êtes un éditeur ?* Ce n'est pas votre numéro de SIRET mais celui de votre client public (qui a effectué la demande d'habilitation) qu'il s'agira de renseigner.

Codes erreurs spécifiques à chaque situation, actionnables et documentés

---

Avec la V3 :
Tous les codes erreur HTTPS sont accompagnés de codes plus précis, spécifiques à chaque situation d'erreur. Une explication en toutes lettres est également donnée dans la payload. Enfin, dans certains cas, une métadonnée actionnable est disponible.
Dans l'exemple ci-dessous, la clé retry_in permet de relancer un appel après le nombre de secondes indiquées.

Exemple de Payload d'un code HTTP 502 :

```
{
  "errors": [
        {
              "code": "04501",
              "title": "Analyse de la situation du compte en cours",
              "detail": "La situation de l'entreprise requiert une analyse manuelle d'un agent de l'URSSAF. Une demande d'analyse vient d'être envoyée, cela prend au maximum 2 jours.",
              "meta": {
                    "provider": "ACOSS",
                    "retry_in": 172800
              }
        }
  ]
}
```

Avant :
Seul le code HTTP standard vous était fourni. Il pouvait correspondre à de nombreuses situations.

Exemple de payload d'un code HTTP 502 :

```
{
  "errors": [
        "L'ACOSS ne peut répondre à votre requête, réessayez ultérieurement  (erreur: Analyse de la situation du compte en cours)"
  ]
}
```

Pourquoi ?

- Pour préciser la nature de l'erreur et vous aider à la comprendre.
- Pour vous permettre d'actionner automatiquement l'erreur en utilisant le code.

Comment ?
Utiliser les libellés pour comprendre l'erreur rencontrée, voire automatiser votre logiciel en fonction du code.
La liste de tous les codes erreurs spécifiques (environ 80) sera ajoutée progressivement au Swagger ainsi qu'à la documentation technique générale.

Volumétrie spécifiée dans le header et actionnable

La gestion de la volumétrie est maintenue identique à la dernière évolution de la V2 et expliquée dans cette documentation.

Gestion des évolutions futures

Avec la V3 :
Toutes les API pourront évoluer indépendamment les unes des autres. Les anciennes versions resteront toujours disponibles.
Le numéro de version devient donc un paramètre de l'appel et non plus une valeur fixe pour toutes les API.

📩Une infolettre annoncera systématiquement les nouvelles évolutions.

Avant :
L'évolution d'un endpoint exigeait la montée en version de toute l'API.

Pourquoi ?

- Permettre l'ajout de nouvelles informations sans forcer les fournisseurs de service à monter de version.
- Continuer de garantir la continuité des API dans le temps.

Comment ?
Renseigner directement le numéro de la version voulue dans l'URL, au même endroit qu'avant, par exemple :
https://entreprise.api.gouv.fr/**v3**/insee/sirene/etablissements/:siret

.

II. Changements relatifs aux API

Un seul fournisseur et un type de données par API

Avec la V3 :
Chaque API appelle un seul et unique fournisseur de données.
Il n'existe plus d'API à contenu multiple, comme celui de l'INSEE, leurs informations ont été découpées en plusieurs API.

Avant :
Certaines API appelaient deux fournisseurs à la fois. Certaines API regroupaient de très nombreuses informations différentes.

Exemple : L'API de V2 "Données de référence d'une entité - /entreprises" est coupé en 3 API dans la V3 :

- Les données Sirene d'une unité légale diffusible (Fournisseur : INSEE)
- Les données Sirene de toutes les unités légales (Fournisseur : INSEE)
- Le siège social d'une unité légale diffusible (Fournisseur : INSEE)
- Le siège social de toutes les unité légales (Fournisseur : INSEE)
- Les mandataires sociaux (Fournisseur : Infogreffe)

Pourquoi ?

- Accélérer le temps de réponse des API, car il y a moins d'appels externes.
- Réduire drastiquement le nombre d'erreurs car le périmètre de la donnée disponible est plus explicite.

Comment ?
Utiliser la table de correspondance pour identifier les nouvelles API.

Les payloads de réponses normalisées et enrichies

Avec la V3 :
La payload de réponse est enrichie de métadonnées actionnables, inscrites dans les clés

“links” et “meta”. Son format est également normalisé à l’aide de la convention JSON API, avec l’ajout d’un identifiant et d’une description de la donnée renvoyée en début de payload.

Les clés “links” renvoient des URL d’appel prêtes à l’emploi permettant d’obtenir des informations supplémentaires. Par exemple, dans l’API renvoyant les données sur une entité légale, un lien est ajouté pour appeler l’API du siège social.

Architecture de la payload :

```
{
“data” : {
        id” : “.....”
        “type” : “.....”,
        “attributes” : {
                LES DONNÉES },
        “links” : {
                ... },
        “meta” : {
                ... }
        }
}
```

Avant :
La structure de la payload n’était pas normalisée ni conventionnée ; elle ne contenait aucune information explicitant la nature de la réponse. Les liens permettant d’appeler une autre API n’étaient pas disponibles.

Architecture la de payload :

```
{
        LES DONNÉES,
}
```

Pourquoi ?

- Uniformiser toutes les payloads de réponses
- Permettre la connexion entre les API grâce à l’ajout des liens URL.